NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

NAVAL WARFARE TACTICAL DATA BASE:
IMPLEMENTATION OF AN INFORMATION
RESOURCE DICTIONARY SYSTEM

by

Bruce Allen Brown
and
Todd Arthur Van Gunten

March 1989

Thesis Advisor: Daniel R. Dolk

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

REPORT SECURITY CLASSIFICATION NCLASSIFIED	1b RESTRICTIVE MARKINGS
SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.
DECLASSIFICATION/DOWNGRADING SCHEDULE	
ERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
NAME OF PERFORMING ORGANIZATION AVAL POSTGRADUATE SCHOOL	6b. OFFICE SYMBOL (If applicable) 54
7a. NAME OF MONITORING ORGANIZATION NAVAL POSTGRADUATE SCHOOL	
ADDRESS (City, State, and ZIP Code) ONTEREY, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code) MONTEREY, CA 93943-5000
NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (If applicable)
9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

TITLE (Include Security Classification)
AVAL WARFARE TACTICAL DATA BASE: IMPLEMENTATION OF AN INFORMATION RESOURCE ICTIONARY SYSTEM

PERSONAL AUTHOR(S)
ROWN, BRUCE ALLEN and VAN GUNTEN, TODD ARTHUR

TYPE OF REPORT	13b TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
ASTER'S THESIS	FROM _____ TO _____	1989, MARCH	110

SUPPLEMENTARY NOTATION THE VIEWS EXPRESSED IN THIS THESIS ARE THOSE OF THE AUTHORS AND DO OT REFLECT THE OFFICIAL POLICY OR POSITION OF THE DEPARTMENT OF DEFENSE OR THE .S. GOVERNEMENT.

COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
NAVAL WARFARE TACTICAL DATA BASE, INFORMATION RESOURCE DICTIONARY SYSTEM, ACTIVE DICTIONARY, PASSIVE DICTIONARY, DATA DICTIONARY

ABSTRACT (Continue on reverse if necessary and identify by block number)

The substantial use of database technology in the collection and dissemination of information for naval tactical systems requires extensive data administration support. The Naval Warfare Tactical Data Base (NWTDB), which is the Navy's authoritative reference for tactical database format and structure, currently lacks sufficient data administration tools to provide an adequate level of data management and control. The feasibility of converting the NWTDB to an Information Resource Dictionary System (IRDS), which will yield an enhanced dictionary capability, is investigated. An analysis of the enhancements generated by the conversion of a passive to active IRDS was also discussed. This research concludes that an IRDS implementation can produce extensive benefits for the NWTDB as well as for other tactical databases in use by the Navy.

DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED	
NAME OF RESPONSIBLE INDIVIDUAL PROF. DANIEL R. DOLK	22b. TELEPHONE (Include Area Code) (408) 646-2260	22c. OFFICE SYMBOL CODE 54 Dk

FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

☆ U.S. Government Printing Office: 1986—606-24.

Approved for public release; distribution is unlimited.

NAVAL WARFARE TACTICAL DATA BASE: IMPLEMENTATION OF AN INFORMATION RESOURCE DICTIONARY SYSTEM

by

Bruce Allen Brown
Lieutenant, United States Navy
B.S., University of Idaho, 1983
and
Todd Arthur Van Gunten
Lieutenant, United States Navy
B.S., United States Naval Academy, 1983

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
March 1989

ABSTRACT

The substantial use of database technology in the collection and dissemination of information for naval tactical systems requires extensive data administration support. The Naval Warfare Tactical Data Base (NWTDB), which is the Navy's authoritative reference for tactical database format and structure, currently lacks sufficient data administration tools to provide an adequate level of data management and control. The feasibility of converting the NWTDB to an Information Resource Dictionary System (IRDS), which will yield an enhanced dictionary capability, investigated. An analysis of the enhancements generated by the conversion of a passive to active IRDS is also discussed. This research concludes that an IRDS implementation can produce extensive benefits for the NWTDB as well as for other tactical databases in use by the Navy.

C.1

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

A. BACKGROUND

Information technologies are playing an increasingly important role in support of DoD missions, programs, and objectives. Database technologies have become an integral component of the Navy's warfighting capability in the tactical as well as the non-tactical arena. Within the last twenty years, organizations have had to learn to manage and control their data resources, but, as the systems get larger and more complicated, they become progressively more difficult to handle. State-of-the-art systems have generated state-of-the-art problems for the Navy to manage, maintain and control.

A database stores and maintains data like a library stores books. This library of data must be structured in such a way that users can access the data easily. All libraries have index catalogs which describe the location of the books. Databases have meta-data which serve the same function. This meta-data is stored in the data dictionary which contains not only the location of the data, but where it is used, how it is used and who uses it. The data dictionary is a very important tool for data administration. When a library receives a new book, it must change the index catalog in many different places; by title, author, subject,

etc. The same type of changes are made when data is added, deleted or changed in a database. Many data dictionaries are passive and require manual updating. An active data dictionary on the other hand is one whose data is automatically modified whenever a change is made to the database.

Controlling and administering a database system is part of the overall data management function and is the responsibility of the Database Administrator (DBA). Management of the data within the database environment involves functions such as database planning and design, access security, and standardization of data elements. Without the necessary tools to accomplish these functions, the DBA cannot be expected to effectively administer the database system in order to realize its potential.

The data dictionary is the primary administrative tool used by the DBA. The basic data dictionary is used for documentation of the data elements within the database. This basic data dictionary can be expanded to accommodate many management and control functions. As will be discussed in detail in the following sections of this thesis, the Information Resource Dictionary System (IRDS) catalogs descriptions of, not only the data elements, but also programs, users, hardware, decision models, or any other information entity deemed important to an organization. The

IRDS increases dictionary capability by including data on essentially the entire information resource environment.

B. OBJECTIVES

The Naval Warfare Tactical Database (NWTDB) is the Navy's authoritative source for characteristic and performance (C&P) data on surface, subsurface and air platforms including their weapons and sensors. It is a centralized database which will eventually hold C&P data on all Red (potentially hostile), Blue (US or allied) and White (neutral) forces, as well as order-of-battle, military installations, digitized maps and charts, weather, oceanography data, merchant ships and cryptologic information [Ref. 1:pp. 2-3]. The primary mission of the NWTDB is to provide the Navy with a single source of tactical information which will eventually be available on a distributed, real time, interactive basis. As this ambitious project matures, its inherent complexity may very well cause significant problems for management. It is vital that project developers plan for this eventuality.

The main thrust of this thesis is to determine the feasibility of integrating the NWTDB into a functional IRDS. This research emphasizes the structure, size and scope of the NWTDB project as it is expected to evolve rather than in its present form. A secondary objective is investigating the implementation of an active IRDS to further enhance the

capabilities and to better support the tasks involved in database administration. An active IRDS could support an integrated database system where different subsystems are controlled and managed under the same administration scheme and provide the interface required for automatic updating and monitoring of the entire organizational information system. If this system can be implemented, not only will the database administration of the present system be enhanced but the growth, maintenance and change within the system will be significantly easier to implement, integrate, control and manage.

C. METHODOLOGY

The first objective of this research is to evaluate and summarize the background information for the NWTDB with particular attention to the administration and management of the project. Secondly, the IRDS model is studied for its potential implementation into the NWTDB system. The third major objective of this research is to evaluate the implications of eventually providing the IRDS with an active capability. The following sources of information were utilized for this effort:

- The Chief of Naval Operations (OP-942) provided the background information on the NWTDB project through applicable instructions, correspondence and policy papers. The history, organization and present status of the project was provided. Current issues as well as future possibilities were discussed via phone conversations and travel to Washington D.C.

- The Naval Intelligence Automation Command (NIAC) provided the technical expertise on the NWTDB project. The database configuration and structure were obtained through the NWTDB User's Guide [Ref. 2] and during a trip to the NIAC facility in Suitland, Maryland.

- The National Bureau of Standards (NSB), which recently approved and promulgated the IRDS standards [Ref. 3], is the main source of information on the IRDS. These standards are analyzed to determine the feasibility and potential benefits of implementation of the IRDS model into the NWTDB. Standards are also in draft [Ref. 4] to convert the IRDS to an active IRDS. The implication of this technology is discussed in the last section of this thesis.

D. THESIS STRUCTURE

The remainder of this thesis is written with the assumption that the reader has a general knowledge of databases and database management concepts. The next section will cover the background information on the NWTDB project. Sections three and four will introduce the IRDS and the Active IRDS models. In the fifth section both models will be applied to the NWTDB system and implementation details will be discussed. The final section of this thesis will provide the reader with recommendations and an overview of research findings. Although this thesis has concentrated on one specific database system, the concepts and methodologies can be translated into database systems Navy wide. We do not cover the cost-benefit analysis which must be the overriding consideration before implementation or modification of any system takes place. This is the responsibility of a system design team.

II. NAVAL WARFARE TACTICAL DATA BASE

A. BACKGROUND

As the Navy progresses in the Era of Information, a major concern is compatibility of database systems. When databases and DBMSs were first introduced in the Navy, each individual activity had its own unique database system, specifically built as a self-contained, stand-alone, independent system [Ref. 1:pp. 1-2]. These activities developed their own unique structures, formats and applications tailored to their organizational and functional requirements. Even database systems developed to be compatible across activities did not remain so for long. These systems go through major changes and modifications during their life cycles. Since there are few if any standardized procedures for maintaining these systems, they quickly become inconsistent and outdated. The Navy soon realized a need for standards, procedures and policies concerning database administration and data management.

Tactical data consistency is of utmost importance. For a Battle Force to successfully counter today's multi-threat environment, the actions of all units must be centrally coordinated and controlled. These conditions require split second analysis and decisions that can only be achieved

using multiple interacting computers. Units must be capable of transmitting and receiving tactical information between their Tactical Data Processors (TDP) without human intervention. Command, Control, Communication and Intelligence (C3I) systems are almost exclusively controlled by digital computers today. Although warfighting capabilities have dramatically improved with the advent of these systems, their full potential will not be realized until they can be effectively integrated. The physical hardware as well as the data itself must be consistent and interchangeable. The integration and interoperability of tactical information systems is a very high priority for the Navy.

The Naval Warfare Tactical Data Base (NWTDB) was developed under direction from the CNO to provide a central repository of tactical information for the Navy. In addition, the CNO directed the NWTDB to be the single authoritative standard for all tactical databases Navy-wide and to support development, evaluation, operations and training in naval tactical warfare [Ref. 5:p. 1]. These standards will facilitate the requirements for database interoperability through definition of standard data elements and formats. The major objectives of the NWTDB are to satisfy the database requirements for [Ref. 6:pp. 1-2]:

- Fleet tactical operations
- Research, development, test and evaluation

- Joint databases in support of multi-service operations
- Combined databases in support of operations with allied forces
- Navy tactical training
- Wargaming

The NWTDB project's standardization effort to date has been in identification and validation of data sources, establishing data element standardization procedures, and development of policies for coordination with tactical database systems that already exist. Interoperability issues, both on the DoN and DoD levels, are now being addressed. Specifically, of high priority now is the incompatibility between intelligence databases and communication protocols.

B. ORGANIZATION AND STRUCTURE

The NWTDB is a branch of the Naval Intelligence Processing System (NIPS) which supports the Navy's requirements for automated tactical intelligence afloat. The NWTDB is located at the Naval Intelligence Automation Command (NIAC), Suitland, Maryland. This program defines tactical database standards, provides for authoritative database fill, and acts as the Navy's focal point for coordinating tactical data base actions with commands and agencies internal and external to the Navy [Ref. 1:p. 3]. NWTDB is the result of merging data from various source databases to create a central repository for tactical

information. These various sources retain responsibility for validation of their respective data and are required to conform to the NWTDB standard for data format and structure [Ref. 5:p. 1].

At the DoD level, the Military Intelligence Integrated Data System/Integrated Data Base (MIIDS/IDB) provides order-of-battle and military installation information for the NIPS. This system was developed under the same standards and will eventually be integrated with NWTDB. MIIDS/IDB is maintained and operated by the Defense Intelligence Agency. The MIIDS/IDB and the NWTDB supply intelligence information to integrated database systems at Theater/Fleet Intelligence Centers for dissemination to fleet users. This is a very simplified description of the NIPS and does not include all the activities that maintain and provide the source data for these systems. What is important to recognize here is the complexity and interdependency this system requires.

It should be apparent that the scope of the NWTDB project is enormous. In fact, the entire program has not been completely defined at this point; the six major functions of the NWTDB which have been identified so far are [Ref. 6:pp. 3-4]:

- Data Element Dictionary (DED), development and dissemination
- Data Collection
- Data Validation
- Data Storage

- Data Maintenance
- Data Dissemination

1. Data Element Dictionary

The DED is used to standardize and document the database elements and structures. The terminology, properties and value ranges of the elements are called meta-data. This meta-data is actually part of the database itself and can be added, deleted and changed exactly like any other data in the database. Figure 1.1 gives an example of a typical entry within the DED.

```
LENGTH_SUB_M     SUBMARINE LENGTH IN METERS

                 5 NUMBER          Picture:  999.9 METERS
Definition:  THE MAXIMUM LENGTH, IN METERS, OF A
SUBMARINE AS MEASURED FROM END TO END.
```

Figure 1.1 Data Element Dictionary Entry

The NWTDB DED is a passive data dictionary and contains the following meta-data for each data element [Ref. 7:p. 2]:

- Element name and short title
- Element type: numeric, character, or date
- Size of the field
- Unit of measure if applicable

- Detailed narrative definition/description of the element
- Miscellaneous attributes:
    - Cognizant agency
    - Source agency
    - Date loaded
    - Date last changed
    - Security classification
    - Control and release markings.

Because every element in the database is identified in the DED, it is an invaluable tool for the administration of the database. It is extremely important that the DED be kept up to date to reflect the constantly changing database. NWTDB data elements are reviewed and conflicts resolved through the OPNAV NWTDB Working Group. Policies originate from the Department of the Navy Information Systems Standards (DONISS) Committee which sets the standards for the Navy's information systems. The DED is contained in the NWTDB User's Guide [Ref. 2].

2. Data Collection

The initial data fill for the NWTDB was collected from the Characteristics and Performance Data Base at the Navy Technical Intelligence Center which contains the C&P data for all Red forces and a limited amount for White forces. Only a small amount of Blue force data has been collected to date. It has been difficult to find an authoritative reference database source for Blue data. Once these data are obtained and validated though, each appropriate OPNAV platform sponsor will be responsible for

maintaining its respective data. As NWTDB is expanded, it will become important for the activities that produce and maintain the source data be able to provide updates to the NWTDB either automatically or at least manually with minimal time delay.

3. Data Validation

All data must be verified and conflicts resolved prior to entering it into the system thus ensuring accurate and authoritative information. Each submitting agency validates and maintains its own data, and has a designated Database Manager who provides quality assurance for the production of validated data. The NWTDB Program Manager resolves data conflicts and has responsibility for validation and verification of data in the NWTDB. The NWTDB Database Administrator and the Data Standards Administrator provide policy and procedure decisions concerning data validation. They also have overall responsibility to ensure data standards are followed. This process of validation and verification is completely manual and very labor intensive.

4. Data Storage

The NWTDB data is stored using the Oracle Relational Database Management System and is structured in two-dimensional tables. The database is grouped into the following data types: platforms, weapons (missiles), weapons (non-missiles), sensors, geographical sites and reference information. These data type groups contain the

data sets as shown in Table 2.1 [Ref. 2:p. 2]. Each data set is described by numerous tables containing relevant characteristics of that particular data set. For instance, the Aircraft Data Set contains all the C&P data available to describe a particular aircraft. Table 2.2 [Ref. 2:p. A1] is a list of the tables located in the Aircraft Data Set. Data sets have a base table and a key attribute which provide logical relationships between the tables internally and externally to that data set. Data set tables contain data elements which are actually attributes of the data set. All data elements from the NWTDB are listed alphabetically in the Data Element Dictionary which has already been explained. Table 2.3 [Ref. 2:p. A1] shows a typical data table from the Aircraft Data Set. Each data element has an associated field type (CHAR), field size (30), and field picture (999.9). Attribute keys, or primary keys as they are referred to here, are denoted by [p]. The NWTDB is maintained on a DEC-VAX 8650 computer with VMS operating system.

5. Data Maintenance

The Database Administrator and Managers are responsible for data maintenance. They provide policies and procedures to support the functionality of the database. As requirements change, the structure and format of the database may also change. Obsolete data will be deleted and

TABLE 2.1

NWTDB DATA SETS

PLATFORMS

1. Aircraft
2. Helicopters
3. Ship Classes
4. Submarine Classes
5. Individual Ships

WEAPONS (MISSILES)

6. Air-to-Air
7. Air-to-Surface
8. Ballistic
9. Surface-to-Air
10. Surface-to-Surface

WEAPONS (NON-MISSILES)

11. Aerial Bombs
12. Aerial Rockets
13. Depth Charges/Bombs
14. Mines
15. Naval Guns
16. Naval Rockets
17. Torpedoes

SENSORS

18. Chaff/IR/RF Decoys
19. Electronic Systems
20. Lasers/Electro-Optics
21. Non-Acoustic Sensors
22. Radars
23. Acoustic Systems

GEOGRAPHICAL SITES

24. Airfields
25. Coastal Defenses
26. Electronic Installations
27. Missile Installations
28. Ports/Anchorages

REFERENCE INFORMATION

29. Data Element Dictionary
30. Reference Tables

TABLE 2.2

AIRCRART DATA SET TABLES

1. ACFT_BASE_TABLE	12. ACFT_NAASW
2. ACFT_USER_CTRY	13. ACFT_AAMS
3. ACFT_PROFILE	14. ACFT_GUNS
4. ACFT_FUNCTIONS	15. ACFT_DEPTH_BOMBS
5. ACFT_POWER_PLANTS	16. ACFT_ASMS
6. ACFT_THREAT_RADII_ELEX	17. ACFT_BOMBS
7. ACFT_THREAT_RADII_WPN	18. ACFT_LASERS
8. ACFT_ELECTRONICS	19. ACFT_AERIAL_RCKTS
9. ACFT_RADARS	20. ACFT_MINES
10. ACFT_SONARS	21. ACFT_TORPEDOES
11. ACFT_CHAFF	

TABLE 2.3

ACFT_GUN DATA TABLE

ACFT_GUNS	
- DESIG_ACFT [p]	CHAR 30
- NAME_ACFT [p]	CHAR 30
- DESIG_ACFT_GUN	CHAR 30
- SIZE_ACFT_GUN	CHAR 10
- TYPE_GUN	CHAR 6
- NBR_ABOARD	999
- NBR_ROUNDS_PER_POD	9999

new data will be added. Maintenance is performed for a variety of reasons [Ref. 8:p. 10]:

- To correct errors and design defects
- To improve the design
- To convert to different hardware, software, system features or telecommunication facilities
- To interface programs
- To make enhancements or necessary changes to the applications

All these functions are maintenance responsibilities. The DED helps the administrator to determine the extent of work needed to implement the changes and how these changes will affect the rest of the database. This information will also allow the administrator to allocate the appropriate resources needed and to inform those individuals who will be affected by the changes. If software and data maintenance can be considered analogous, corrective, adaptive, perfective, and preventive maintenance costs can be expected to range from 40 to 70 percent of the administrative budget [Ref. 9:p. 529]. The Air Force can attest to development costs of $75 per instruction while maintenance costs run as high as $4000 per instruction [Ref. 8:p. 14].

6. Data Dissemination

Presently NWTDB can only be accessed through the local network on which it resides. Output for subscribers is limited to magnetic tape, cartridges, diskettes and hardcopy reports. Requests for data or inquiries to the

database are processed via written requests. As mentioned before, tactical databases today require near-real-time, interactive processing in order to support the tactical environment. These databases must be capable of transmitting and receiving tactical information via telecommunication links. Since they were originally designed to operate in a stand-alone capacity, connecting these systems electronically will not be as simple as connecting a transmit/receive device. The major problem still lies in data element compatibility. Data format and structure, procedures, hardware, and communication media must all be standardized. These problems are in the process of being identified and resolved in order to make NWTDB readily available to the fleet.

C. DATA STANDARDIZATION

A major priority for the NWTDB project is tactical data standardization. Timeliness and accuracy of tactical data can only be achieved if database systems can communicate. DoD policies concerning data element standardization are contained in DOD DIRECTIVE 5000.11 [Ref. 10]. This directive requires conformance to the following guidances when applicable:

- Federal Information Processing Standards (FIPS)
- International Standard Organization (ISO)

- North Atlantic Treaty Organization (NATO)
- American National Standards Institute (ANSI).

Many DoD policies concerning data standardization were written before requirements of the C3I environment were known. Only recently has DoD began to revise these outdated policies. Because of the insufficient guidance that existed, several lower level efforts to standardize data elements and communication formats emerged in an attempt to fill the gap.

Defense Intelligence Agency (DIA) initiated the Intelligence Data Element Authorized Standards (IDEAS) which is utilized extensively in MIIDS/IDB and NWTDB. To support the joint arena, JCS sponsored the Joint Interoperability of Command and Control Systems (JINTACCS) program which standardizes formats for bit-oriented (tactical data links) and character-oriented messages (message text format). JCS Pub 25 specifies these formats which the National Security Agency (NSA) also utilizes for fleet cryptologic support. Over-the-Horizon Targeting (OTH-T) GOLD Format (OTG) is another standardized format for contact reporting between Tactical Data Processors supporting OTH-T systems. Naval warfare communication and database format standardization policies [Ref. 11] specify that all tactical naval warfare database systems will conform to NWTDB format and all systems communicating externally will use JINTACCS and/or OTH-T GOLD formats.

NWTDB C&P data is principally produced under the auspices of DIA. As a consequence, MIIDS/IDB and NWTDB is closely comparable. At present though, NWTDB standards are not totally consistent with JINTACCS and OTH-T GOLD standards. Ongoing coordination to resolve these inconsistencies is continuing through the NWTDB Working Group and the DONISS Committee. These efforts only address compatibility at the Navy tactical level and do little for the theater and national levels. Coordination is needed to resolve conflicts among data element standards for databases, data links and external communications. DoD and DoN information systems must be interoperable to support today's C3I environment. Until these issues are resolved, conflicting data standards will adversely effect tactical warfighting capabilities, particularly in joint operations.

D. FUTURE DEVELOPMENTS

NIAC has spent close to $5 million on the NWTDB program through FY-89 and another $500,000 is planned for FY-90. It is uncertain with the budgetary constraints, how much support and growth the program can expect in the future. Besides the standardization efforts and the continued support for the NWTDB database itself, there are a few ambitious projects being discussed. The Navy plans to eventually link all C3I data processing systems to Department of Defense Intelligence Information System

(DODIIS) telecommunications. This will provide remote users near-real-time access for updating and validation of tactical information. The ultimate goal is to provide the fleet with a single C3I tactical reference database which is accurate, timely, consistent and interoperable. To this end, a proposal has also been made for integrating all C3I information systems. This system will be called the Tactical Data Integration System (TDIS) and will contain not only platform C&P, weapons and sensor data, but information on:

- Communication
- Counter-measures
- Navigation aids
- Order-of-battle
- Military installations
- Acoustic and electronic parameters/signatures
- Cryptology
- Meteorology
- Oceanography
- Topography
- Warfare doctrine

NWTDB will evolve as requirements and technology change. It will need to be continually restructured to meet new demands, new tactical threats, and inevitable conflicts with other data systems. If information systems were static, then the data administration supporting them could afford to

be static. This is not the case though. Innovative and dynamic data administration is essential to the effectiveness of information systems. A dynamic and flexible approach is needed to assure the NWTDB is able to [Ref. 10:pp. 3-4]:

- maintain a current, on-line data base of tactically relevant information which is traceable to producers
- provide tactical data to support Fleet operations, training, test and evaluation, and wargaming
- provide coordination between tactical users, system developers and producers of NWTDB data fill
- support on-line access by users and producers who have access to high volume data transfer communications (e.g., DODIIS)
- provide the final quality control check for accuracy and consistency of data and format.

The increasing complexity and corresponding growth of information systems will undoubtedly require a different approach to managing them. Taking advantage of automated management tools available for data administration may be the only way to have a fighting chance.

III. INFORMATION RESOURCE DICTIONARY SYSTEM

A. INTRODUCTION

In many organizations the largest portion of a database manager's operating budget is spent on data maintenance. The maintenance of existing software ranges from 40 to 70 percent of an operating budget [Ref. 12:p. 525]. If a large organization has numerous database management systems it may also have many different data dictionary systems. Since the data dictionary system is the primary software tool used to manage data it seems reasonable to suggest that a standardized data dictionary system should reduce data maintenance costs. In 1980, both the American National Standards Institute (ANSI) and the National Bureau of Standards (NBS) recognized this opportunity to develop an expanded data dictionary system. This data dictionary system, which will be used as a standard throughout the Federal Government, is called the Information Resource Dictionary System (IRDS).

Before development began on the IRDS a cost-benefit analysis was performed by the Institute of Computer Sciences and Technology (ICST) at NBS. The purpose of the study was to estimate the cost savings resulting from the implementation of an IRDS standard. The study concluded that by the early 1990's the government could expect over

$120 million in benefits from the use of this new dictionary system. The study specifically identified four major opportunities for cost reduction and avoidance [Ref. 3:p. 4]:

- Improved identification of existing, valuable information resources that can be used by others in the same organization or shared with other organizations.
- Reductions of unnecessary development of computer programs when suitable programs already exist.
- Simplified software and data conversion through the provision of consistent documentation.
- Increased portability of acquired skills resulting in reduced personnel training costs.

The IRDS will be used to develop, modify, and maintain manual and automated database systems, to support an organization-defined data element standardization program, and to support records, reports, and forms management, ranging from non-automated to fully automated environments.

B. DESIGN OBJECTIVES

An important phase of any project development is the identification of specific design objectives. Precise objectives were established for the proper development of the IRDS. ANSI and the Institute for Computer Sciences and Technology (ICST), a separate department of the National Bureau of Standards, jointly identified three major objectives [Ref. 3:p. 5]:

- The IRDS should contain the major features and capabilities that exist in currently available dictionary systems.

- The IRDS should be modularized to support a wide range of user environments and to support cost-effective procurement.
- The IRDS should support portability of skills and data.

The first design objective ensured that current and projected technology for existing dictionary systems were included in the IRDS. Components that performed necessary functions were incorporated into the IRDS standard.

The second design objective established a modularized structure for the IRDS. The structure consists of a "Core" plus three additional modules. The Core contains the basic capabilities that are currently found in "state-of-the-art" dictionary systems. The purpose of the three supplementary modules are to provide an increased level of security, an application program interface, and database management system documentation support.

The third and final design objective was to create a system that would support portability of skills and data. This objective ensures that personnel who currently use dictionary systems will have few problems in converting to the IRDS. The IRDS supports two user interfaces. The first is a menu-driven panel which allows the inexperienced user to perform tasks by using menus. The second interface is a Command Language which requires more sophistication and will probably be used by more experienced personnel. Even though the Command Language is more difficult to use, it is a quicker and more efficient way of operating the IRDS. The

final objective also ensures that data can be exchanged from one IRDS to another. Since all IRD Systems are standardized the transfer of data between systems can be accomplished easily.

C. IRDS ARCHITECTURE

The structure of the IRDS consists of four basic layers. The lowest or first layer of the IRDS structure is the database itself. This layer contains instances of entities and relationships that describe the real world. The function of this layer is to store data. The second layer of the IRDS structure is IRD data. This layer contains entities, relationships, and attributes. The function of this layer is to describe data contained in the database. The third layer of the IRDS structure is IRD schema. This layer contains entity-types, relationship-types, and attribute-types. The function of this layer is to describe and control the IRD layer. The highest and final layer of the IRDS structure is the IRD schema description. This layer contains meta-entities, meta-relationships, and meta-attributes. The function of this layer is to describe and control the IRD schema layer. Figure 3.1 presents the structure of the IRDS.

The entire IRDS is based on the notion of entities, relationships, and attributes. An entity represents or describes a "real world" concept, person, event, or

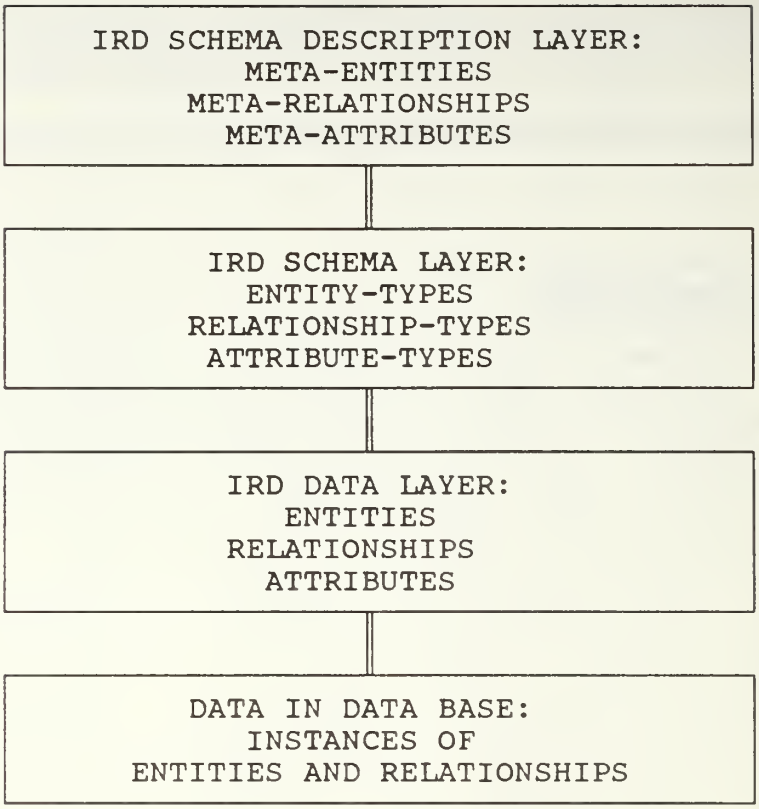

Figure 3.1 Information Resource Dictionary System

quantity, but is not the actual data that exists in an application file or database. A relationship is simply an association between two entities, and an attribute represents properties of an entity or a relationship [Ref. 3:p. 9].

The IRDS is based on binary relationships, that is a relationship can occur only between two entities. The core does not allow for more than a one to one relationship. The strategy of using binary relationships was chosen for two reasons:

- Most current implementations of data dictionary systems use the binary approach.
- Binary relationships simplify the task of implementation.

This four layer framework comprises the basic structure of the IRDS. An example of typical IRDS contents is provided in Figure 3.2. A more detailed examination of the IRD schema layer and the IRD schema description layer is needed to fully understand the IRDS standard.

D. IRD SCHEMA

As previously stated, the IRD schema describes and controls the structure of the IRD. For each entity, relationship, and attribute that exists in the IRD, the IRD schema contains a corresponding entity-type, relationship-type, and attribute-type [Ref. 3:p. 12]. To standardize all IRD systems, a unique set of entity-types, relationship-types, and attribute-types were established. This group of standard types is called the Core System-Standard Schema.

The Core System-Standard Schema consists of twelve distinct entity-types grouped into three different categories: data, process, and external (Table 3.1) [Ref. 13:pp. 16-17]. Data entity-types define how data is structured in the information resource environment. For example, consider the case where we have a data element ALT_MAX_FT (maximum altitude of an aircraft) which is part of a file AIRCRAFT. ALT_MAX_FT and AIRCRAFT are entities of

IRD SCHEMA DESCRIPTION LAYER	META-ENTITY TYPES		
	ENTITY-TYPES	RELATIONSHIP-TYPES	ATTRIBUTE-TYPES
IRD SCHEMA LAYER	FILE ELEMENT	FILE CONTAINS ELEMENT	LENGTH
IRD LAYER	AIRCRAFT FILE AIRCRAFT NAME	AIRCRAFT_FILE CONTAINS AIRCRAFT_NAME	30 CHARACTERS
DATA IN DATABASE	U.S. AIRCRAFT TOMCAT	AIRCRAFT_FILE FOR U.S._AIRCRAFT CONTAINS [TOMCAT]	(ATTRIBUTES DO NOT APPEAR IN THE DATABASE)

Figure 3.2 Contents of a Typical IRDS

TABLE 3.1a

CATEGORY I: DATA ENTITY-TYPES

1. DOCUMENT. This entity-type is used to describe instances of human readable data collections. Examples are reports and forms.

2. FILE. This entity-type is used to describe instances of an organization's data collections. Examples of different files could be Aircraft-File and Ship_Classes-File.

3. RECORD. This entity-type is used to describe instances of logically associated data that belongs to an organization. Examples of different records could be Aircraft-Record and Ship_Classes-Record.

4. ELEMENT. This entity-type is used to describe instances of data belonging to an organization. Examples of different elements could be Aircraft-Name and Ship_Classes-Name.

5. BIT_STRING. This entity-type describes abstract representations of strings of binary digits.

6. CHARACTER_STRING. This entity-type describes abstract representations of strings of characters.

7. FIXED_POINT. This entity-type describes abstract representations of exact numeric values.

8. FLOAT. This entity-type describes abstract representations of approximate numeric values.

Instances of the bit-string, character-string, fixed-point, and float do not directly represent entities, but are used by relationships to describe characteristics of ELEMENT.

TABLE 3.1b

CATEGORY II: PROCESS ENTITY-TYPES

9. SYSTEM. This entity-type describes instances of collections of processes and data.
10. PROGRAM. This entity-type describes instances of automated processes.
11. MODULE. This entity-type describes instances of automated processes that are either logical subdivisions of PROGRAM entities or independent processes that are called by PROGRAM entities.

TABLE 3.1c

CATEGORY III: EXTERNAL ENTITY-TYPES

12. USER. This entity-type describes individuals or organizational components.

type ELEMENT and FILE respectively. Further they participate in the relationship AIRCRAFT-CONTAINS-ALT_MAX_FT which is of type FILE-CONTAINS-ELEMENT. Process entity-types define operations that manipulate data in the information resource environment. A SYSTEM entity-type might contain a MODULE or PROGRAM entity-type whose purpose

is to perform an output function that produces a list of files contained in a database. User entity-types define the participants of the information resource environment. A participant may be a individual or an organization component. Records department and financial department are examples of organizational components.

This method of categorizing entity-types into data, process, and user groups helps define the information resource environment. Categorizing entity-types is a concept not found in previous data dictionary systems. Most relational database management systems combine all entities into one group. Separating entity-types into groups enhances the level of description in the IRD schema layer.

The Core System-Standard Schema consists of eight unique relationship-types [Ref. 3:pp. 18-19]. These relationship-types are designed to capture the important associations between entity-types in the information resource environment. Table 3.2 presents the relationship-types contained in the Core System-Standard Schema.

Unlike entity-types and relationship-types, attribute-types for the Core System-Standard Schema are not specifically identified. Organizations develop unique attributes-types that are appropriate for their system. Some attribute-types are common to all IRDS entity-types and provide audit trail information and general documentation for the schema [Ref. 3:pp. 16-19]. Table 3.3 presents the

TABLE 3.2

RELATIONSHIP-TYPES

1. CONTAINS. This relationship-type describes instances of an entity being composed of other entities. An example of this relationship is SYSTEM-CONTAINS-PROGRAM.

2. PROCESSES. This relationship-type describes the association between PROCESS and DATA entities. An example of this relationship is USER-PROCESSES-FILE.

3. RESPONSIBLE_FOR. This relationship-type describes the association between entities representing organizational components and other entities, to denote organizational responsibility. An example of this relationship is USER-RESPONSIBLE_FOR-RECORD.

4. RUNS. This relationship-type describes the association between USER and PROCESS entities. An example of this relationship is USER-RUNS-SYSTEM.

5. GOES_TO. This relationship-type describes the "flow" association between PROCESS entities. An example of this relationship is SYSTEM-GOES_TO-SYSTEM.

6. DERIVED_FROM. This relationship-type describes the association between entities where a target entity is the result of a calculation involving a source entity. An example of this of relationship is RECORD-DERIVED_FROM-FILE.

7. CALLS. This relationship-type describes the "calling" association between PROCESS entities. An example of this relationship is PROGRAM-CALLS-MODULE.

8. REPRESENTED_AS. This relationship-type describes the association between ELEMENT and certain other entities that document the ELEMENT's format. An example of this relationship is ELEMENT-REPRESENTED_AS-CHARACTER_STRING.

TABLE 3.3

COMMON ATTRIBUTE-TYPES

1. ACCESS_NAME. This attribute-type provides each entity with a unique identifying name. This name is used as the primary method of identifying an entity.
2. DESCRIPTIVE_NAME. This attribute-type provides each entity with a unique descriptive name. This name is usually more detailed and meaningful than ACCESS_NAME.
3. ALTERNATE_NAME. This attribute-type provides each entity with an alternate name. This name is usually a "synonym" or "alias" for ACCESS_NAME.

common attribute-types contained in the Core System-Standard Schema. An example of these common attribute-types might be the use of NAME_ACFT as the access name, AIRCRAFT_NAME as the descriptive name, and PLANE_NAME as the alternate name. These identifying names provide access to the IRD.

These unique entity-types, relationship-types, and attribute-types constitute the standardized structure of the IRD schema. All IRD Systems must conform to this standard, however additional entity-types, relationship-types, and attribute-types can be added to the schema. The capability to modify the structure of the schema allows an organization to customize its IRDS.

E. IRD SCHEMA DESCRIPTION

The purpose of the IRD schema description layer is to describe and control the IRD schema layer. Entity-types, relationship-types, and attribute-types presented in the previous section are defined in terms of meta-entities. These meta-entities are linked by meta-relationships, and both meta-entities and meta-relationships can have meta-attributes associated with them. In the same manner in which the schema describes entities, relationships, and attributes in the IRD, the schema itself is described using meta-entities, meta-relationships, and meta-attributes [Ref. 3:p. 53]. Even though the IRD schema description layer is similar in structure to the IRD schema layer, the IRD schema description layer consists of a unique set of meta-entities, meta-relationships, and meta-attributes that constitute the IRDS standard.

This unique set of meta-entities exists in the schema description layer and employs similar terminology used in the schema layer when naming meta-entities. Entity-types, relationship-types, and attribute-types that were found in the schema layer can also be found in the schema description layer as meta-entities. The schema description layer also includes the following new entities [Ref. 3:pp. 53-54]:

- RELATIONSHIP-CLASS-TYPE
- ATTRIBUTE-GROUP-TYPE
- ATTRIBUTE-TYPE-VALIDATION-PROCEDURE

        - RANGE VALIDATION
        - VALUE VALIDATION

- ATTRIBUTE-TYPE-VALIDATION-DATA

        - (DEFINED BY ORGANIZATION)

- VARIATION-NAMES-DATA

        - (DEFINED BY ORGANIZATION)

- LIFE-CYCLE-PHASE

        - UNCONTROLLED-PHASE
        - CONTROLLED-PHASE
        - ARCHIVED-PHASE
        - SECURITY-PHASE

- QUALITY-INDICATOR

        - (DEFINED BY ORGANIZATION)

- SCHEMA-DEFAULTS

        - EXISTING-SCHEMA-DEFAULTS

This distinct set of meta-entities helps to describe the structure of the schema layer. The meta-entity level information describes the logical structure of the IRD schema layer for the instances that will be stored at the IRD layer [Ref. 13:p. 52].

Meta-relationships are used in the schema description layer to describe associations between meta-entities. Similar to the IRD schema layer, the IRD schema description layer uses the binary approach in representing a meta-relationship. For each association between two meta-entities there exists only one meta-relationship. Figure 3.3 displays the similarities between a relationship in

the schema layer and a relationship in the schema description layer.

Figure 3.3a A Relationship in the IRD Schema Description Layer

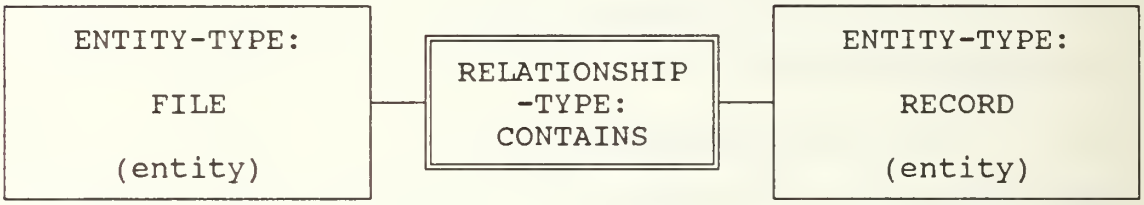

Figure 3.3b A Relationship in the IRD Schema Layer

In the schema description layer, meta-attributes are used to describe meta-entities and meta-relationships. These meta-attributes are categorized by function into four different groups. The following are the four groups of meta-attributes [Ref. 3:pp. 56-58]:

- DOCUMENTATION META-ATTRIBUTES. These meta-attributes are used for documenting the purpose of the meta-entity.
- AUDIT META-ATTRIBUTES. These meta-attributes are used for recording audit information.

- SCHEMA CONTROL META-ATTRIBUTES. These meta-attributes provide control over the schema and determine what can or cannot be done.
- DICTIONARY CONTROL META-ATTRIBUTES. These meta-attributes are used to impose rules on the dictionary.

These distinctive meta-entities, meta-relationships, and meta-attributes constitute the standardized structure of the schema description layer. Each individual IRDS will use the necessary meta-data to describe its own schema layer. The contents of the schema description layer will depend on the make up of the schema layer.

F. FUNCTIONS AND FACILITIES OF THE IRDS CORE

As previously discussed, the IRDS will be used for the development, modification, and maintenance of database systems. It will also be used to support records, reports, and forms management. The IRDS contains specific functions and facilities which allow for the accomplishment of these tasks. The following will be discussed in this section:

- Populating and Maintaining the IRD.
- Manipulating and Redefining the IRD Schema.
- The Dictionary Output Facility.
- The IRD-IRD Interface Facility.
- The Procedure Facility.
- The IRDS Control Facilities.

These functions and facilities are considered part of the Core of the IRDS.

The IRDS possesses the ability to populate and maintain the IRD. This function can create, modify, and delete entities and relationships in the dictionary. This function also has the ability to create a new entity with the same attributes and relationships as an existing entity. This procedure is known as "copying an entity." If an AIR_TO_SURFACE_MISSILE entity has the same attributes and relationships as an AIR_TO_AIR_MISSILE entity, the "copying an entity" function can be used to create the AIR_TO_AIR_MISSILE entity if the AIR_TO_SURFACE_MISSILE entity already exists. This unique ability to change the structure of the dictionary gives the IRDS the flexibility to adapt as the database management system changes.

The IRDS also has the ability to manipulate and redefine the IRD schema. This function can create, modify, and delete meta-entities and meta-relationships in the schema description layer. Further, this function can produce a generalized output of the schema contents. The ability to retrieve and modify information about the structure of the schema gives the IRDS the flexibility to adapt as the IRD changes.

The IRDS contains a dictionary output facility that is used to produce the following [Ref. 3:pp. 41-45]:

- General Output. This process produces a user defined list of dictionary entities, their associated relationships, and the attributes of these entities and relationships. Example of output in Figure 3.4a.

```
ENTITY:
AIRCRAFT_FILE

RELATIONSHIPS:
AIRCRAFT_FILE-CONTAINS-AIRCRAFT_NAME
AIRCRAFT_FILE-CONTAINS-TYPE_ACFT
AIRCRAFT_FILE-CONTAINS-WT_PAYLOAD_KG
AIRCRAFT_FILE-CONTAINS-SPD_ACFT_MAX_MACH
RECORD_DEPARTMENT-RESPONSIBLE_FOR-AIRCRAFT_FILE

ATTRIBUTES:
ACCESS_NAME
DESCRIPTIVE_NAME
ALTERNATE_NAME
LAST_MODIFIED


[ Relationships and attributes of AIRCRAFT_FILE ]
```

Figure 3.4a General Output

- Impact-of-Change Output. This process will be used to identify those entities that might be affected by a change to a specified entity. Example of output in Figure 3.4b.

- Syntax Output. This process will generate a list of command language instructions that were necessary to create a selected entity. Example of output in Figure 3.4c.

- Entity-Lists Output. This process will generate a user defined list of access-names. Example of output in Figure 3.4d.

The dictionary output facility allows a user to quickly and easily obtain important information about the IRD.

The IRDS also contains an IRD-IRD interface facility which is used to transfer data between dictionaries. This

```
ENTITY TO BE MODIFIED:
AIRCRAFT_NAME

ENTITIES AFFECTED BY MODIFICATION:
AIRCRAFT_FILE
AIRCRAFT_FUNCTION
AIRCRAFT_POWER_PLANTS

  [ Entities affected by modification of entity
              AIRCRAFT_NAME ]
```

Figure 3.4b Impact-of-Change Output

```
INSERT INTO ENT_TYPE VALUES

('NAME_ACFT', 'AIRCRAFT_NAME',
         'PLANE_NAME', 'J.J. DOE', '10JAN89')


[ Command Language instruction needed to create the
              NAME_ACFT entity ]
```

Figure 3.4c Syntax Output

```
ENTITIES:
NAME_ACFT
TYPE_ACFT
WT_PAYLOAD_KG
SPD_AIRCRAFT_MAX_MACH


  [ Entities that represent the ELEMENTS of the
              AIRCRAFT_FILE ]
```

Figure 3.4d Entity-Lists Output

interface transfers entities and relationships from one IRD to another. In order for this transfer to take place, both dictionaries must conform to the IRDS standard. If these dictionaries are not compatible the interface facility will not allow the transfer. The IRD-IRD interface facility provides the capability of sharing important IRD information with many users.

The procedure interface enables the IRDS to save a series of operations that were used to create entity-lists or IRD output. This "series of operations" is known as a procedure. Saved procedures can be retrieved and executed as they are needed. The ability to save procedures allows for quick and easy recovery of entity-lists or IRD output.

Lastly, the IRDS core consists of five control facilities. These facilities are as follows [Ref. 13:p. 52]:

- Versioning Facility. This process provides every entity with an assigned version-identifier. The version-identifier in turn links the entity with its associated revision. As entities change, the versioning facility provides a means of identifying the most current revision of an entity.

- Life-Cycle-Phase Facility. This process assigns each entity to a life-cycle-phase. The life-cycle-phase establishes integrity rules for the IRD. A dictionary system has different life-cycles-phases and unique entities associated with each particular life-cycle. This facility controls the movement of these unique entities from one life-cycle phase to another.

- Quality-Indicators Facility. This process provides every entity with an assigned quality-indicator. The quality indicator denotes the level of standardization of element entities and the degree to which the entity satisfies an organization's Quality Assurance or Quality

Testing methodology. This facility is used to indicate the level of development an entity such as designed, coded, and tested.

- Views. This process defines a user's access to different segments of the IRDS.

- Core Security. This process designates which user has access permission to enter the IRD and the IRD schema.

Each facility provides a unique method of control and are used in populating and maintaining the Information Resource Dictionary.

G. FUNCTIONS AND FACILITIES OF THE IRDS MODULES

As previously described, the IRDS consists of a "Core" plus three additional modules. Each module performs a unique function for the IRDS. These modules comprise the basic IRDS structure [Ref. 3:pp. 105-109]:

- Entity Level Security Module. This module provides the IRDS with the capability of assigning "read" and "write" privileges for individual entities. A user must have the proper access key to either view or modify an entity. This increased level of security enables a system to limit a user from having the capability of modifying the structure of the IRDS.

- Application Program Interface Module. This module provides the capability of interfacing a standard programming language with the IRDS. A program, written in a standard programming language, will have the ability to extract information from the IRDS.

- Support of Standard Data Models Module. This module provides the IRDS with the capability of supporting Natural Database Language (NDL) and Standard Query Language (SQL) databases. Entity-types, relationship-types, and attribute-types are use to extend the IRDS schema in order to integrate the NDL and SQL databases.

This modularized structure is the basic framework of the IRDS standard.

H. CONCLUSION

The standardized model presented here is only the basic structure of the Information Resource Dictionary System. Each individual information system will modify the structure of the IRDS to conform to its specific needs. The IRDS model presented in this chapter may also be extended to an active data dictionary system. This enhancement provides greater capabilities for interfacing the IRDS to existing software systems which use the meta-data which the IRDS contains.

IV. THE ACTIVE IRDS

A. OVERVIEW

The American National Standards Institute (ANSI) has extended the original capabilities of the IRDS by issuance of the IRDS Services Interface standard. The original standards as discussed in the previous chapter did not provide the capability for programs and systems external to the IRDS to communicate directly and interactively with the IRDS. The core IRD-IRD interface facility only allows the transfer of data between two dictionaries conforming to the IRDS standard. This prevents the IRDS from supporting an integrated information resource management environment. The many software tools available to management, such as teleprocessing systems, Decision Support Systems, Computer Aided Software Engineering (CASE) tools, and database management systems, would each have to maintain its own proprietary data and data structures. This situation could cause extensive data redundancy and maintenance problems not to mention the inability to centrally control and manage the overall system.

The IRDS Services Interface standard defines a program-based interface to the IRDS that allows programs external to an IRDS to populate, access, and maintain the contents of an IRDS dictionary and its schema [Ref. 4]. With this

interface, the IRDS can be used actively as a database of information resource descriptions for any external program or system that provides a call mechanism and supports character, integer, and real data types. These external systems can store and maintain a wide variety of information resources within the IRDS including applications, programs, files, databases, data elements, hardware devices, and documents. The IRDS Services Interface standard contains the same modules as the original IRDS standard except for the Basic Functional Schema and the Entity Lists, modules 2 and 7 respectfully. This standard has completely duplicated the functionality of the original standard and only requires the implementation of Module 1, the Core Standard, as the original standard does. All other modules are optional. This chapter discusses important aspects of this standard and provides the reader a general overview of the IRDS Services Interface. Most of the information has been obtained from the standard itself and the reader is referred to the ANSI X3.nnn-1988 IRDS Services Interface publication [Ref. 4] for further implementation details.

The IRDS Services Interface standard has not changed the basic structure of the IRDS as described in the previous chapter. The IRDS is still based on the Entity-Relationship (E-R) Model and consists of the four main levels: Database, IRD, IRD Schema and the IRD Schema Definition levels. As mentioned before, the IRDS Services Interface standard

extends the capabilities of the basic IRDS by introducing new concepts and facilities which build upon the basic IRDS structure. The basic IRDS has already been covered extensively in Chapter III.

B. CONCEPTS AND SERVICES

The IRDS Services Interface standard uses Pascal as the specification language and an implementation of this standard will require as a minimum, one of the following languages; FORTRAN, COBOL, PL/1, C, PASCAL or ADA. The source code for the Services Interface will be maintained internally through the IRDS and external users will be required to provide a method for invoking the source code program and linking their application to the IRDS. An implementation conforming to this standard must also adhere to the constants, data types, service protocols, IRD and IRD Schema definitions as contained in this standard.

The standard provides for user-defined extensions to all major data structure definitions. The IRDS is a fully extensible dictionary system and by adding or deleting meta-entities, meta-relationships and meta-attributes the user can change the IRD Schema and in the process redefine the IRD structure. This will enable users to modify the IRDS to respond to new and changing requirements and thus support their own unique operating environment.

Services are provided to both the IRD and the IRD Schema levels but users are normally allowed to access only the IRD level. Since there is very little difference between the IRD structure and the IRD Schema structure it is no longer necessary to make the distinction. For the purposes of this paper, the IRD and IRD Schema will be referred to as the E-R structure unless specifically stated otherwise.

An external application invokes the IRDS with a service call. The application will then initiate an IRDS session by invoking the OPEN_IRDS service protocol. This opens the E-R structure for modification. When the update is completed, the session is ended with a CLOSE_IRDS service protocol. Each session is divided into transactions. A transaction can be considered an update to the IRDS. During a transaction, the update can be saved or cancelled. This is accomplished by the COMMIT and ROLLBACK services which will make an update permanent or return it to its original state respectfully. The CLOSE_IRDS service will terminate the session and will end the current transaction with an implied COMMIT.

The size of an IRDS transaction is consistent with the size of the application task being processed. The transaction size is therefore an application design consideration. Since any object being used in a transaction prevents another application from using it, a small transaction size will likely increase system throughput.

Also in the case of application or system failure, the user would not lose as much of the current process if the transaction size were small. Large transactions may also increase the likelihood of a deadlock situation occurring. This situation occurs when two applications are each holding the object the other application is attempting to retrieve. This stand-off situation prevents either of the applications from completing its transaction. The IRDS deals with deadlock by automatically invoking the ROLLBACK service on one of the applications involved and allows the other to proceed.

Objects are retrieved from the IRDS by associating the object with what is referred to as a cursor. A cursor is used to establish position on the object. The E-R structure can only be entered by opening a cursor and establishing its position on an entity. It is not possible to enter directly through a relationship or attribute. Once cursor position has been established on an entity, any relationship or attribute in which this entity participates can then be accessed. In other words, the E-R structure is navigated by positioning the cursor on the "source" entity, then traversing its associated relationship to finally gain access to the "target" entity. Since the E-R structure is maintained in a sequential order, all entities, relationships and attributes can be hierarchically accessed. Entities can be added from anywhere within the E-R structure

but relationships and attributes can only be added by positioning on an associated entity. Also to modify or delete any object, cursor position on that object must first be established. This ensures the integrity of the E-R structure.

Although the user is unaware of it, the E-R structure is not accessed directly. An application accesses a subset of the E-R structure called the subschema. This protects the application from any changes made to the E-R structure while the application is processing. If a modification is made to the E-R structure which invalidates the application, the application would then be interrupted and recompiled immediately to reflect the change.

When a cursor is opened, the user or application specifies the key to the object set desired. A set corresponds to the row within a database table to which the target object belongs. Selection of objects by keys is the most common retrieval method used although objects can also be retrieved based upon values such as attributes or quality indicators. Also objects are usually returned in sequence, but if specified, can be returned in any order desired.

Entities can be accessed by one of two keys: Access Name or Descriptive Name. The Access Name is an abbreviated name whereas the Descriptive Name is an extended version providing for better clarification. For example, ships in the Navy are grouped according to classes. The entity for

classes may have an Access Name such as CLASS with a Descriptive Name of SHIP_CLASSES. Both of these keys can have the following components:

- Assigned Access Name
- Version Identifier
- Revision Number.

The primary component of an access key is the Assigned Access Name which is the actual name of the entity. The Version Identifier is used to describe a variation of the same entity. For example, the Navy decided to redesignate a class of ships from Destroyer Escorts (DE_CLASS) to Fast Frigates (FF_CLASS). FF_CLASS is not a new entity, just a variation of the original entity DE_CLASS. The Revision Number is a positive integer which is automatically incremented as new revisions of an entity are created.

As previously discussed, relationships can only be accessed via an associated entity. In the IRDS, relationships must be binary and directed. Binary means that only one-to-one relationships are allowed and directed means that one of the participating entities is superior to the other. Besides establishing position on the entity, the direction of the relationship must also be specified. When traversing from a superior entity to a subordinate entity, the Relationship Direction is said to be forward (CLASS-CONTAINS-FF_CLASS). Going the other way, the Relationship

Direction is said to be inverse (FF_CLASS-CONTAINED_IN-CLASS). Once position is established and direction is specified, the target entity can then be determined. A combination of these specifications is considered the Relationship Access Key. By evaluation of these relationships, the IRDS also utilizes this portion of the E-R structure to perform the Impact-of-Change function. This function will identify the effect that any change to the E-R structure has on the rest of the structure.

As was the case with relationships, a plural attribute is accessed by first establishing position on a participating entity. A single plural attribute is identified by the value of the attribute. For a plural attribute group, the value of the significant attribute within the group will be the key for identifying all attributes for that group.

Figure 4.1 expands our example of SHIP_CLASSES and will be utilized here as a simplistic illustration of how the IRDS is accessed and navigated. Say for instance, a user needs to modify an attribute within the IRD level. The application opens the IRDS and enters the IRD level. Position is established on the entity CLASSES by the Open-Entity-Cursor procedure. The relationship CLASSES-CONTAINS-FF_CLASS must then be traversed. By specifying the direction (FORWARD) and the target entity (FF_CLASS), position is established on the entity FF_CLASS. The

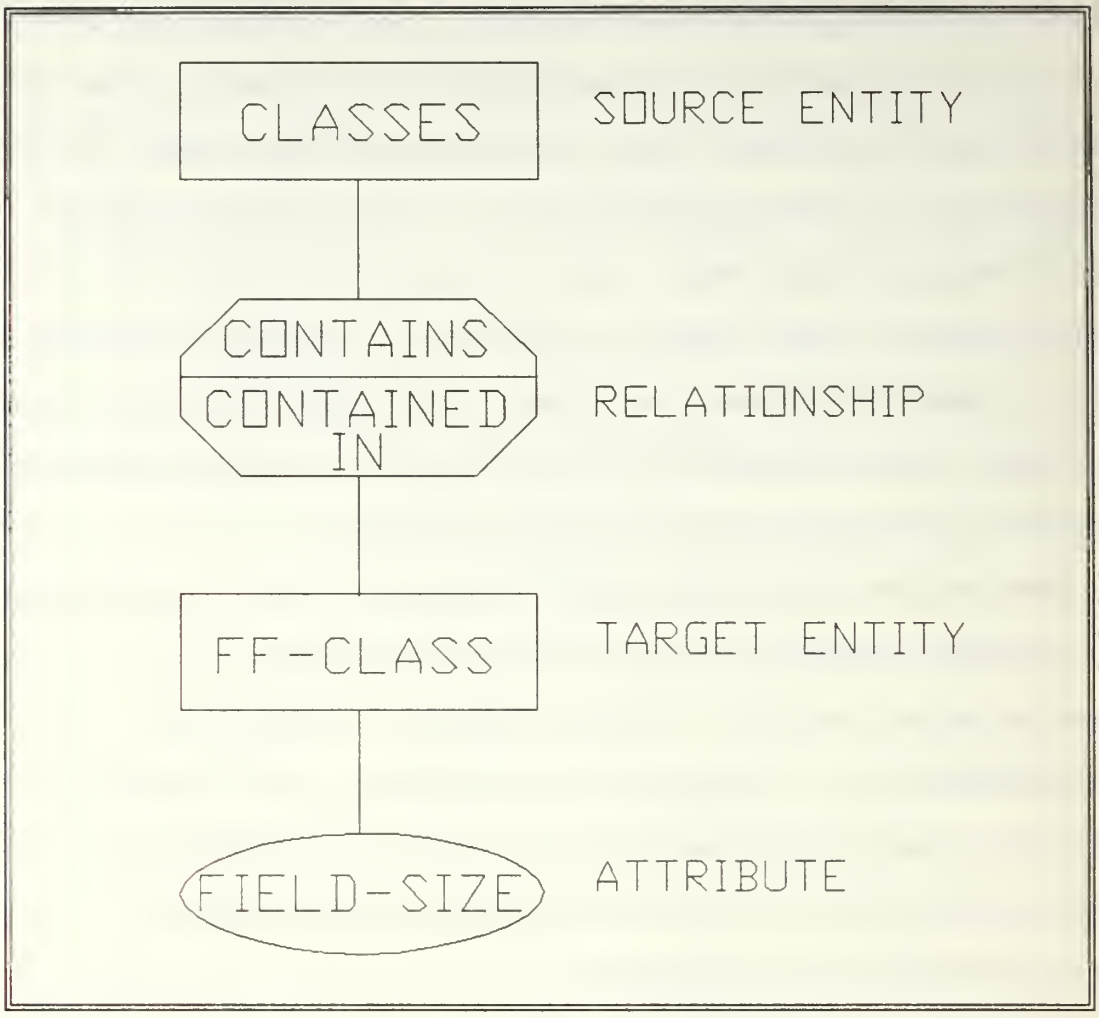

Figure 4.1 IRDS Access and Navigation Illustration

attribute FIELD_SIZE is now accessible and modification to this attribute can be accomplished by specifying the type of service to be conducted, in this case the Modify-Attribute procedure.

Access keys for entities, relationships and attributes are contained in IRDS records. Besides the access key,

entity records contain audit attributes, life cycle phase and quality indicators. Relationship records specify the relationship type and target entity. Plural attribute records include just the attribute type. These records also contain variant fields to which the IRDS application is sensitive. A null field will indicate if there is a value present for a variant.

As mentioned before, all entities can have associated audit and quality indicator attributes. The audit attributes enhance management and control of entity modification occurrences. They are as follows:

- ADDED_BY
- DATE_TIME_ADDED
- LAST_MODIFIED_BY
- DATE_TIME_LAST_MODIFIED
- NUMBER_OF_TIMES_MODIFIED

The quality indicator supplements the life cycle phase facility as a measure of an entity's overall quality. For example, the life cycle phase will indicate that a certain entity is in the development stage while the quality indicator will indicate the level of development such as designed, coded, tested, etc.

The IRDS Services Interface standard provides a complete list of all constants and data types required for defining the various IRDS services. Table 4.1 is a list of procedures necessary to perform IRDS services. These

TABLE 4.1

IRDS SERVICE PROCEDURES

OPERATIONAL SERVICES	
Open IRDS	Commit
Close IRDS	Rollback
Retrieve Error Message Line	Set Session Defaults

IRD SCHEMA LEVEL SERVICES	IRD LEVEL SERVICES
Open Meta-Entity Cursor	Open Entity Cursor
Close Meta-Entity Cursor	Close Entity Cursor
Retrieve Meta-Entity	Retrieve Entity
Modify Meta-Entity	Modify Entity
Delete Meta-Entity	Delete Entity
Add Meta-Entity	Add Entity
Open Meta-Relationship Cursor	Open Relationship Cursor
Close Meta-Relationship Cursor	Close Relationship Cursor
Retrieve Meta-Relationship	Retrieve Relationship
Modify Meta-Relationship	Modify Relationship
Delete Meta-Relationship	Delete Relationship
Add Meta-Relationship	Add Relationship
Open Meta-Attribute Cursor	Open Attribute Cursor
Close Meta-Attribute Cursor	Close Attribute Cursor
Retrieve Meta-Attribute	Retrieve Attribute
Modify Meta-Attribute	Modify Attribute
Delete Meta-Attribute	Delete Attribute
Add Meta-Attribute	Add Attribute
Modify Access Name	Get Attribute Type
Modify Descriptive Name	Decode Attribute Value
Modify Life Cycle Phase	Modify Access Name
Copy Meta-Entity	Modify Descriptive Name
Deactivate IRD	Modify Life Cycle Phase
Install IRD Structure	Copy Entity
Restore IRD Schema Activate IRD	

service procedures are provided by the IRDS and are accessible to users through external applications. In this way, the IRDS can be accessed, navigated and maintained by

any external application containing the appropriate call mechanism.

When an IRDS service is completed or if at any time an error occurs, the application will receive one of the following return codes along with a reason code providing specific details of what has happened to the service:

- SERVICE SUCCESS: indicates service was completed successfully.
- SERVICE_FAILURE: indicates service was not completed because of a failure not due to user or application.
- SERVICE_ERROR: indicates service was not completed due to user or application error.
- TRANSACTION_ERROR: indicates current transaction has failed.
- IRDS_ERROR: indicates IRDS Services Interface has failed.

C. EXTENSIBLE LIFE CYCLE PHASE FACILITY

The E-R structure is subdivided into logical partitions. These partitions separate all related entities, life cycle phases and security layers. The life cycle phase partitions separate definitions according to their level of completeness. The three life cycle phases are:

- UNCONTROLLED: further development or testing needed
- CONTROLLED: fully developed and tested
- ARCHIVED: historically significant, no longer used.

The Core Standard IRDS provides for life cycle management of entities within the IRD. The Extensible Life Cycle Phase Facility is an attempt to strengthen the life

cycle control and management facilities of the IRD Schema. This facility extends the capabilities of the IRD Schema Definition level by the addition of the following enhancements to the IRD schema structure:

- A mechanism for defining life cycle phase integrity.
- The Install IRD Structure Service for the movement of meta-entities to and from the life cycle phases.
- A documentation aid for the IRD schema.
- A method for establishing a hierarchy of IRD life cycle phases.

D. IRDS SECURITY MODULE

There are also partitions in the E-R structure for IRDS security. Each user of the system has an assigned access identifier defined in the security partition under the entity type IRDS_USER. The IRDS can be accessed only if the user's access identifier matches the IRDS_USER entity type. Access is denied if no match exists. The IRDS_USER entity is related to the IRDS_VIEW entity. These entities control the user's capabilities in accessing the different levels within the IRDS. The IRDS Services Interface standard specifies two layers of security as illustrated in Figure 4.2. The first layer is the Global Security Facility which applies to both the IRD and IRD Schema levels. This layer is considered "closed" which means access is granted only if explicitly authorized. The Global Security layer contains a

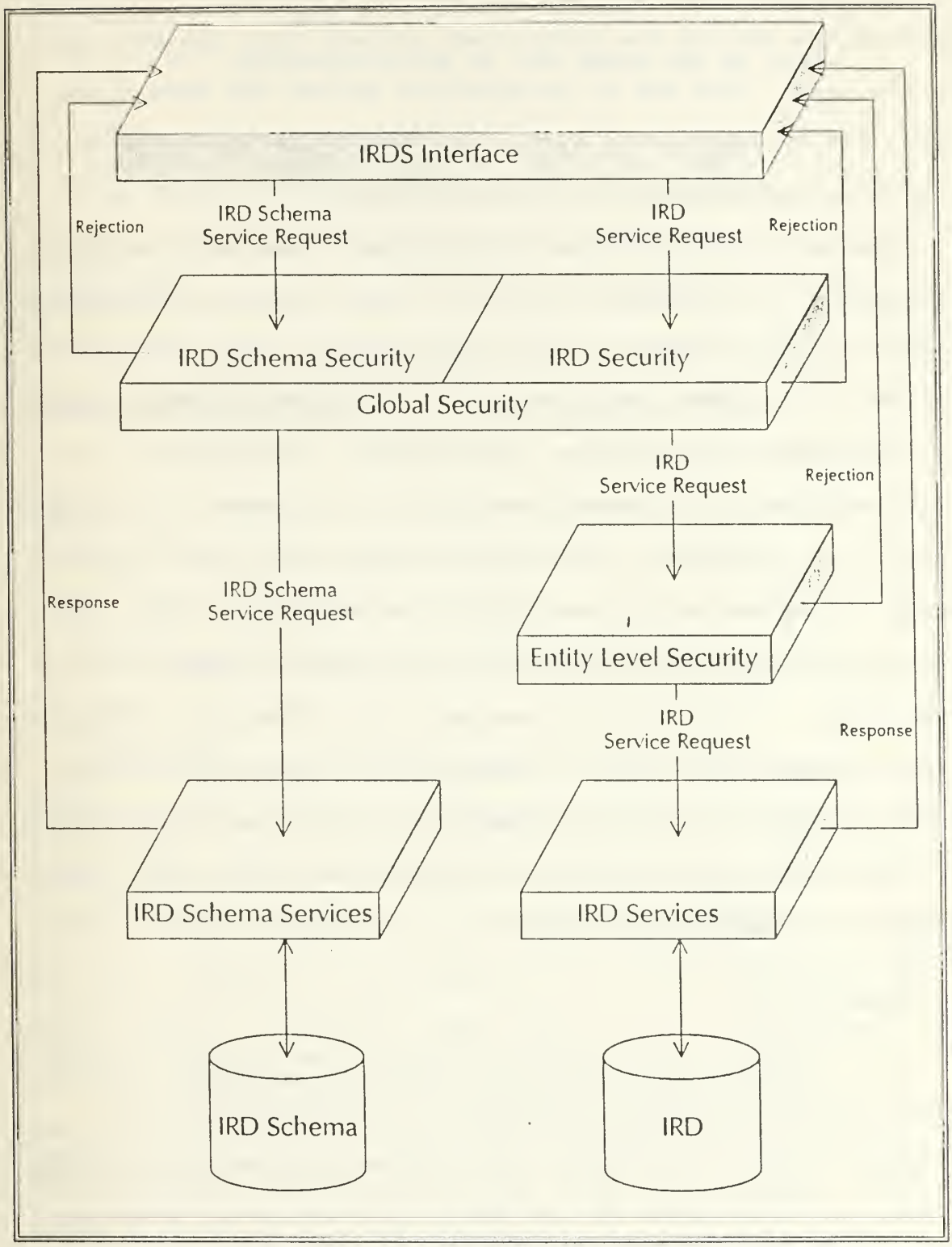

Figure 4.2 IRDS Security Overview

set of permissions (READ, ADD, MODIFY, DELETE, etc.). These permissions define the authorized actions that can be taken by the user. The set of permissions define the IRDS_VIEW. Views can be associated with either IRDS partitions or as attributes of entities of type IRDS_USER.

The second layer is the Entity Level Security Facility and applies to just the IRD level. This layer is considered "open" in that access is authorized unless explicitly denied. The Entity Level Security Facility restricts access to individual IRD entities. An entity is considered restricted when an Access Controller is attached. An Access Controller has one of two locks, a read lock, or a write lock. A locked entity can only be accessed with the appropriate access key which is attached to the user's IRDS_VIEW. A user with a read key will be able to read but cannot modify the entity. A user with a write key will be able to read and modify the entity. Secured entities appear as if they do not exist to all users except those with the appropriate access key.

E. CONCLUSION

This chapter has surveyed the important aspects of the IRDS Services Interface standard. In conjunction with Chapter III, these concepts will now be applied to the NWTDB system and will serve as the basis upon which our eventual recommendations will be built.

V. IMPLEMENTATION OF THE NWTDB AS AN IRDS

A. NWTDB DATA DICTIONARY

The NWTDB, which was designed and developed as a performance oriented database management system, does not adequately support the administrative aspects of information management. This system sufficiently defines the basic data resource environment but cannot provide a description of the overall information environments in which the NWTDB will eventually operate. The implementation of the NWTDB as an IRDS will provide a more comprehensive dictionary capability, a substantial foundation for future additions of metadata support tools, and compliance with a federal standard [Ref 13:p. 49].

The NWTDB dictionary currently lacks the necessary structure and data that are essential for the implementation of an IRDS. This dictionary presently consists of a one layer logical structure that contains information describing the data elements of the NWTDB. The implementation of the IRDS requires a distinct three layer logical structure: the IRD, the IRD schema, and the IRD schema description. Further, the NWTDB dictionary only contains information describing the data element portion of the database. The NWTDB dictionary must be expanded to include information

describing the entire structure of this database as well as environments in which the NWTDB may eventually be used.

B. IMPLEMENTATION OF THE IRD LAYER

The NWTDB dictionary layer, which presently consists of a data element dictionary, must be expanded to include information about records, tables, files, modules, programs, systems, documents, users and relationships that are contained in, or presently use, the NWTDB. For each instance of an entity and relationship in the NWTDB, IRD data must be created. Figure 5.1 presents the basic implementation structure for each instance of a relationship in the NWTDB [Ref. 13:p. 54]. Figure 5.2 presents the basic implementation structure for each instance of an entity in the NWTDB [Ref. 13:p. 54].

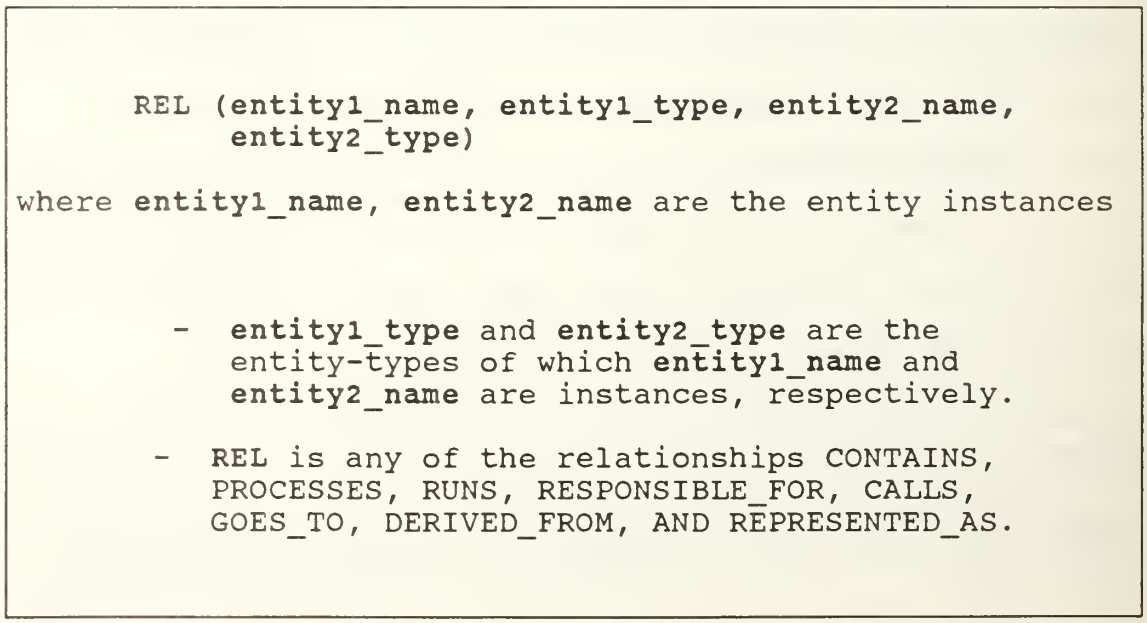

Figure 5.1 Relationship Structure of the IRD Layer

```
SYSTEM (access_name, descriptive_name, added_by,
        dated_added, modified_by, date_modified,
        modification_number, duration_value,
        duration_type, comments, description,
        security_classification)

PROGRAM (access_name, descriptive_name, added_by,
         date_added, modified_by, date_modified,
         modification_number, duration_value,
         duration_type, language, lines_of_code,
         comments, description, security_classification)

MODULE (access_name, descriptive_name, added_by,
        date_added, modified_by, date_modified,
        modification_number, duration_value,
        duration_type, language, lines_of_code,
        comments, description, security_classification)

  FILE (access_name, descriptive_name, added_by,
        date_added, modified_by, date_modified,
        modification_number, number_of_tables,
        number_of_records, comments, description,
        security_classification)

 TABLE (access_name, descriptive_name, added_by,
        date_added, modified_by, date_modified,
        modification_number, number_of_records,
        comments, description, security_classification)

RECORD (access_name, descriptive_name, added_by,
        date_added, modified_by, date_modified,
        modification_number, record_category,
        comments, description, security_classification)

ELEMENT (access_name, descriptive_name, added_by,
         data_added, modified_by, date_modified,
         modification_number, data_classification,
         low_range, high_range, comments, description,
         security_classification)

DOCUMENT (access_name, descriptive_name, added_by,
          date_added, modified_by, date_modified,
          modification_number, document_category,
          comments, description, security_classification)

  USER (access_name, descriptive_name, added_by,
        date_added, modified_by, date_modified,
        modification_number, comments, description,
        security_classification)
```

Figure 5.2 Entity Structure of the IRD Layer

The entity structure of the IRD follows the basic architecture of the IRDS standard with one minor exception. In order to define the structure of the NWTDB, the IRD has been modified to include information about tables.

1. Element Data

Element data of the IRD consists of information pertaining to a standardized set of attributes. These unique attributes are used to describe each element of the database. The standardized attributes include the access name of the element, the descriptive name of the element, the user who added the element, the date the element was added, the user who last modified the element, the date the element was modified, the number of modifications to that element, the element's data classification, the high-range of the element, the low-range of the element, explanatory comments about that element, a description of the element, and the element's security classification. For every element of the NWTDB, IRD data must be created. A typical example of IRD data, as it relates to attributes describing elements, is presented in Figure 5.3.

Some of the element attributes that exist currently in the NWTDB's dictionary data are identical to the standardized attributes in the IRD. These existing NWTDB attributes can be converted to IRD attributes by just renaming them. One example of this conversion is that

```
ELEMENT (access_name, descriptive_name, added_by,
         date_added, modified_by, date_modified,
         modification_number, data_classification,
         low_range, high_range, comments, description,
         security_classification)


ELEMENT ('NAME_ACFT', 'AIRCRAFT NAME', 'B.A. Brown'
         '10 Oct 86', 'T.A. Van Gunten', '23 Feb 89'
         '10', 'Character', '1', '30',
         'Soviet Aircraft', 'Name of Aircraft',
         'Unclass')
```

Figure 5.3 IRD Data for NAME_ACFT Element

MNEMONIC_ELEMENT, an attribute that presently exists in the NWTDB dictionary, can be changed to ACCESS_ NAME. This alteration allows for existing NWTDB dictionary data to be converted into IRD dictionary data. Additionally, dissimilar NWTDB attributes can be combined with required IRD attributes, thus creating a new attribute structure for the IRD. These modifications are dependent upon the needs and desires of the implementors and can be integrated into the attribute structure of the IRD, as long as the required set of core attributes remain unchanged.

Every data element in the IRD, with the exception of those defining the data dictionary core attributes, will be identical with those used in the current NWTDB dictionary.

Annex B of the Naval Warfare Tactical Database User's Guide identifies each element of the database [Ref. 2]. Elements employed to describe the structure of the data dictionary will conform to the finalized attribute structure of the IRD.

2. Record Data

Record data of an IRD also contains a standardized set of attributes. These unique attributes are used to describe each record of the database. The standardized attributes include the access name of the record, the descriptive name of the record, the user who added the record, the date the record was added, the user who last modified the record, the date the record was modified, the number of modifications to that record, the record category, explanatory comments about the record, a description of the record, and the record's security classification. For every record in the NWTDB, IRD record data must be created. Unlike IRD element data, the conversion of NWTDB dictionary data into IRD dictionary data is not possible. This conversion is not possible because current record data does not exist in the NWTDB.

In order to identify a record in the NWTDB, specific elements of the database are designated as primary keys. For each instance of a designated primary key, there exists a uniquely identified record. This distinct record is associated with one instance of a primary key and numerous

instances of associated elements. In the NWTDB, an example of a primary key is DESIG_ACFT. Instances of DESIG_ACFT would include F-14, F-4, A-6, and any other aircraft associated with ACFT file. This primary key is used to associate instances of NAME_ACFT, TYPE_ACFT, IOC, IOC_RMKS, CTRY_CODE_MFGD, WT_PAYLOAD_MAX_KG, and several other elements with an individual record. Each of these unique records, F-14, F-4, A-6, and any other aircraft associated with ACFT file, are associated with a specific set of attributes and relationships which define the record structure of that table and file.

3. Table Data

Table data of an IRD has the same basic structure as element data and record data. The standardized set of attributes for table data uniquely describe each table of the NWTDB. The standardized attributes include the access name of the table, the descriptive name of the table, the user who added the table, the date the table was added, the user who last modified the table, the date the table was modified, the number modifications to the table, the number of records in the table, explanatory comments about that table, a description of the table, and the table's security classification. For each table in the NWTDB, IRD table data must be created.

The NWTDB consists of thirty different files and each of these files contains a certain number of tables.

Each file contains a base table, which is considered to be the primary table, and additional tables that are logically subordinate to the base table. An example of table structure in the NWTDB is presented in Figure 5.4.

FILE:	Aircraft Data Set
BASE TABLE:	Acft
ADDITIONAL TABLES:	
Acft User Ctry	Acft Functions
Acft Profile	Acft Threat Radii Elex
Acft Power Plants	Acft Threat Radii Wpn
Acft Electronics	Acft Radars
Acft Sonars	Acft Chaff
Acft NAASW	Acft AAMS
Acft AAMS	Acft ASMS
Acft Guns	Acft Aerial Rckts
Acft Bombs	Acft Mines
Acft Lasers	Acft Depth Bombs
Acft Torpedoes	

Figure 5.4 Table Structure in the NWTDB

4. File Data

File data is created by using a standardized set of attributes that uniquely defines each file of the NWTDB. The standardized attributes include the access name of the file, the descriptive name of the file, the user who added the file, the date the file was added, the user who last modified the file, the date the file was modified, the number of modifications to the file, the number of tables in

the file, the number of records in the file, explanatory comments about the file, a description of the file, and the file's security classification. As previously stated, the NWTDB consists of thirty files. For each of these files, IRD data must be created. Figure 5.5 provides a complete listing of NWTDB files.

FILES OF THE NWTDB

AIRCRAFT	HELICOPTERS
SHIP CLASSES	SUBMARINE CLASSES
INDIVIDUAL SHIPS	AIR-TO-SURFACE MISSILES
AIR-TO-AIR MISSILES	BALLISTIC MISSILES
SURFACE-TO-AIR MISSILES	SURFACE-TO-SURFACE MISSILES
AERIAL BOMBS	AERIAL ROCKETS
DEPTH CHARGES/BOMBS	MINES
NAVAL GUNS	NAVAL ROCKETS
TORPEDOES	CHAFF/IR/RF DECOYS
ELECTRONIC SYSTEMS	LASERS/ELECTRO-OPTICS
NON-ACOUSTIC SENSORS	RADARS
ACOUSTIC SENSORS	AIRFIELDS
COASTAL DEFENSES	ELECTRONIC INSTALLATIONS
MISSILE INSTALLATIONS	PORTS/ANCHORAGES
REFERENCE TABLES	IRD

Figure 5.5 Files of the NWTDB

5. System Data

Unlike the implementation of element, record, table, and file data, the implementation of system data will be a relatively easy undertaking. In fact, the system data describing the NWTDB is the only system data included in the IRD. As the NWTDB becomes integrated and networked, system

data will added to the IRD for every additional system associated with this database.

System data is created by using a standardized set of attributes that uniquely defines each system. The standardized attributes include the access name of the system, the descriptive name of the system, the user who added the system, the date the system was added, the user who last modified the system, the date the system was modified, the number of modifications, the duration value of the system, the duration type system, explanatory comments about the system, a description of the system, and the system's security classification.

6. Program Data

Program data is generated by using a standardized set of attributes that uniquely defines each program of the NWTDB. The standardized attributes include the access name of the program, the descriptive name of the program, the user who added the program, the date the program was added, the user who last modified the program, the date the program was modified, the number of modifications to the program, the duration value of the program, the duration type of the program, the language the program is written in, the length of the program in lines of code, explanatory comments about the program, a description of the program, and the program's security classification.

Programs that create, manipulate, or delete any portion of NWTDB's data or structure must be included in the IRD. Programs used to produce output listings must also be included in the dictionary. For every program contained in or used by the NWTDB, IRD data must be created.

7. Module Data

Most programs are constructed using modular design. For each function a program performs, a separate module is produced. For every module identified as being a part of a NWTDB program, IRD data must be created.

Module data will be generated by using a standardized set of attributes that uniquely defines each module of the NWTDB. The standardized attributes include the access name of the module, the descriptive name of the module, the user who added the module, the date the module was added, the user who last modified the module, the date the module was modified, the number of modifications to the module, the duration value of the module, the duration type of the module, the language the module was written in, the length of the module in lines of code, explanatory comments about the module, a description of the module, and the module's security classification.

8. Document Data

Document data will be created by using a standardized set of attributes that uniquely defines each document of the NWTDB. The standardized attributes include

the access name of the document, the descriptive name of the document, the user who added the document, the date the document was added, the user who last modified the document, the date the document was modified, the number of modifications to the document, the document category, explanatory comments about the document, a description of the document, and the document's security classification.

For every report and form that is either imported for or generated by the NWTDB, IRD document data must be created. Examples of currently existing NWTDB documents include the All-Data Hardcopy Report and the Text File Printout.

9. User Data

A user is defined as any individual or organization component that is associated with the NWTDB. The specific name of an individual, the particular name of a division, the unique name of a department, or the name of an affiliated activity can be employed to identify users of the NWTDB. For each of these users, IRD data must be created.

User data will be created by using a standardized set of attributes that uniquely defines each user of the NWTDB. The standardized attributes include the access name of the user, the descriptive name of the user, the individual who added the user, the date the user was added, the individual who last modified user data, the date of the modification, the number of modifications, explanatory

comments about the user, a description of the user, the location of the user, and the user's security classification (Figure 5.6).

```
USER (access_name, descriptive_name, added_by,
      date_added, modified_by, date_modified,
      modification_number, comments, description,
      security_classification)

USER ('NAVSEA', 'Naval Sea Command', 'NIAC',
      '27 Dec 85', 'null', 'null', 'null'
      'Can only be modified by NAIC',
      'User of NWTDB', 'Unclass')
```

Figure 5.6 IRD Data for NAVSEA User

10. Relationship Data

Relationship data for the IRD is created by using a standardized set of attributes that uniquely defines each important association in the NWTDB. In accordance with Figure 5.1, the standardized attributes include the name of the first associated entity, the first entity's type, the name of the second associated entity, and the second entity's type.

IRD data will be generated for every instance of a CONTAINS, RUNS, CALLS, RESPONSIBLE_FOR, PROCESSES, DERIVED_FROM, GOES_TO, and REPRESENTED_AS relationship that exists in the NWTDB. Each of these newly implemented relationships must conform to one of the required IRD

integrity constraints. Figure 5.7 presents a complete listing of relationship constraints for the IRD layer [Ref. 13:p. 52]. A sampling of different IRD relationships are presented in Figure 5.8. Once this relationship data has been entered, the implementation of the IRD layer is complete.

C. IMPLEMENTATION OF THE IRD SCHEMA LAYER

The IRD layer uses entities, relationships, and attributes to uniquely define instances of entities and relationships in the NWTDB. The IRD schema layer employs entity-types, relationship-types, and attribute-types to categorize each instance of an entity, relationship, and attribute of the IRD layer.

Every instance of an element, record, table, file, module, program, system, document, and user entity in the IRD layer must be designated as one of the following IRD schema layer entity-types: ELEMENT, RECORD, TABLE, FILE, MODULE, PROGRAM, SYSTEM, DOCUMENT, and USER. For each of these entity-types, IRD schema description metadata must be created.

Every instance of an attribute in the IRD layer must be designated as one of the following IRD schema layer attribute-types: ACCESS_NAME, ACCESS_METHOD, ADDED_BY, ALLOWABLE_VALUE, SECURITY_CLASSIFICATION, CODE_LIST_LOCATION, COMMENTS, DATA_CLASS, DATE_ADDED,

```
CONTAINS (system,system)        PROCESSES (system,file)
CONTAINS (system,program)       PROCESSES (system,table)
CONTAINS (system,module)        PROCESSES (system,record)
CONTAINS (program,program)      PROCESSES (system,element)
CONTAINS (program,module)       PROCESSES (system,document)
CONTAINS (module,module)        PROCESSES (program,file)
CONTAINS (file,file)            PROCESSES (program,table)
CONTAINS (file,document)        PROCESSES (program,record)
CONTAINS (file,table)           PROCESSES (program,element)
CONTAINS (file,record)          PROCESSES (program,document)
CONTAINS (file,element)         PROCESSES (module,file)
CONTAINS (table,table)          PROCESSES (module,table)
CONTAINS (table,record)         PROCESSES (module,record)
CONTAINS (table,element)        PROCESSES (module,element)
CONTAINS (record,record)        PROCESSES (module,document)
CONTAINS (record,element)       PROCESSES (user,file)
CONTAINS (element,element)      PROCESSES (user,table)
CONTAINS (document,document)    PROCESSES (user,record)
CONTAINS (document,record)      PROCESSES (user,element)
CONTAINS (document,element)     PROCESSES (user,document)

RESP_FOR (user,file)        DERIVED_FROM (document,file)
RESP_FOR (user,table)       DERIVED_FROM (document,table)
RESP_FOR (user,record)      DERIVED_FROM (document,record)
RESP_FOR (user,element)     DERIVED_FROM (document,element)
RESP_FOR (user,document)    DERIVED_FROM (document,document)
RESP_FOR (user,system)      DERIVED_FROM (element,file)
RESP_FOR (user,program)     DERIVED_FROM (element,table)
RESP_FOR (user,module)      DERIVED_FROM (element,record)
                            DERIVED_FROM (element,element)
RUNS (user,system)          DERIVED_FROM (element,document)
RUNS (user,program)         DERIVED_FROM (record,file)
RUNS (user,module)          DERIVED_FROM (record,table)
                            DERIVED_FROM (record,record)
CALLS (module,module)       DERIVED_FROM (record,document)
CALLS (program,program)     DERIVED_FROM (table,file)
CALLS (program,module)      DERIVED_FROM (table,table)
                            DERIVED_FROM (table,document)
GOES_TO (system,system)     DERIVED_FROM (file,file)
GOES_TO (program,program)   DERIVED_FROM (file,document)
GOES_TO (module,module)
```

Figure 5.7 Relationship Constraints of the IRD Layer

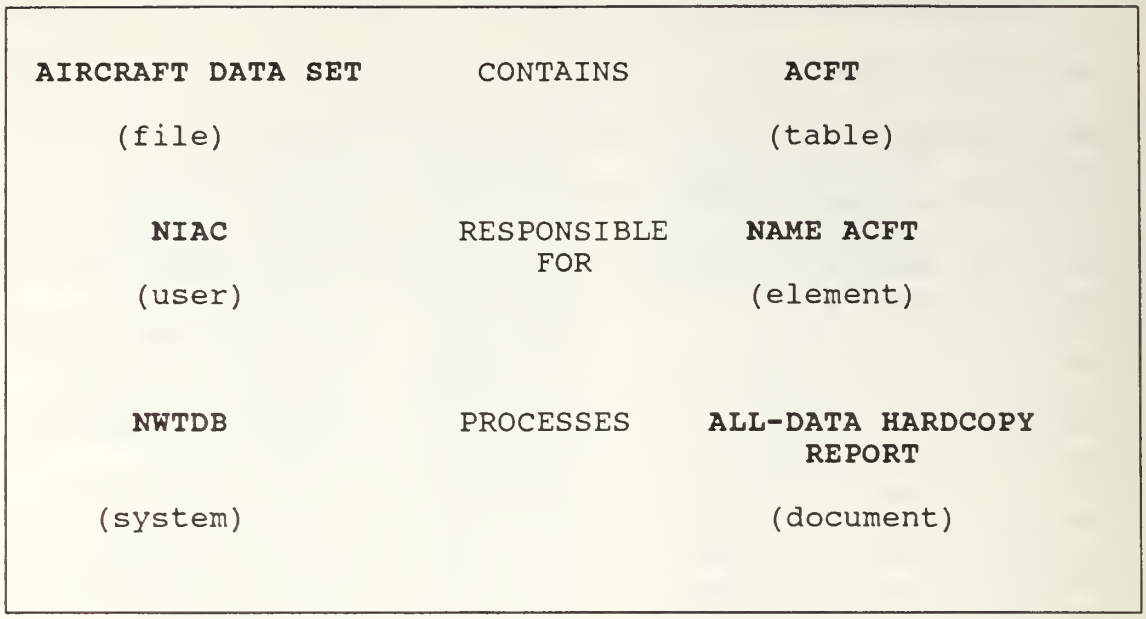

Figure 5.8 Sample Relationships of the IRD

DESCRIPTION, DESCRIPTIVE_NAME, DOCUMENT_CATEGORY, DURATION_TYPE, DURATION_VALUE, FREQUENCY, HIGH_RANGE, LAST_ MODIFICATION_DATE, LAST_MODIFIED_BY, LOCATION, LOW_RANGE, LINES_OF_ CODE, MODIFICATION_NUMBER, NUMBER_ OF_RECORDS, NUMBER_OF_TABLES, RECORD_CATEGORY, RELATIVE_POSITION, and SECURITY. For each of these attribute-types, IRD schema description metadata must be created.

Every instance of a relationship in the IRD layer must be designated as one of the following IRD schema layer relationship-types: CONTAINS, PROCESSES, RUNS, RESPONSIBLE_FOR, GOES_TO, CALLS, DERIVED_FROM, and REPRESENTED_AS. For each of these relationship-types, IRD schema description metadata must also be created.

D. IMPLEMENTATION OF THE IRD SCHEMA DESCRIPTION LAYER

As previously described, the IRD schema layer employs entity-types, relationship-types, and attribute-types to categorize each instance of an entity, relationship, and attribute of the IRD layer. The IRD schema description layer is used to describe the logical structure of that IRD schema. For each instance of an entity-type, attribute-type, and relationship-type in the IRD schema layer, IRD schema description metadata must be created. Figure 5.9 presents the basic implementation structure for each instance of a meta-entity in the IRD schema layer [Ref. 13:p. 55]. In Figure 5.9, meta-entity types are represented in uppercase and the meta-attributes of each meta-entity type are represented in lowercase. A typical example of IRD schema description metadata, as it relates to meta-attributes describing meta-entities, is presented in Figure 5.10.

Meta-attributes of the IRD schema description layer are used to describe meta-entities and meta-relationships. As discussed in Chapter III, the meta-attributes are categorized into four groups: documentation, audit, schema control, and dictionary control. This implementation scheme uses access name, descriptive name, added by, dated added,

```
ENT_TYPE (access_name, descriptive_name, added_by,
          date_added, modified_by, date_modified,
          modification_number, comments, description,
          security_classification)

ATT_TYPE (access_name, descriptive_name, added_by,
          date_added, modified_by, date_modified,
          modification_number, comments, description,
          security_classification)

REL_TYPE (access_name, descriptive_name, added_by,
          date_added, modified_by, date_modified,
          modification_number, comments, description,
          security_classification)
```

Figure 5.9 Meta-Entity Structure
of the IRD Schema Description Layer

```
ENT_TYPE  (access_name, descriptive_name, added_by,
           date_added, modified_by, date_modified,
           modification_number, comments, description,
           security_classification)



ENT_TYPE  ('SYSTEM', 'System Entity-Type', 'D.A. Jones',
           '24 Mar 83', 'null', 'null', 'null',
           'entity-type of IRD schema layer',
           'system entity-type of the IRD schema layer',
           'unclass')
```

Figure 5.10 Example of IRD Schema Description Metadata

modification number, comments, description, and security classification as meta-attributes to describe each meta-entity. The addition of new meta-attributes to the IRD schema description layer will be dependent on the needs and desires of the of the implementors. Further, the IRD schema description layer implementation scheme uses the identical attribute structure of the IRD layer to describe each meta-relationship of the IRD schema description layer.

Meta-relationship data of the IRD schema layer can be created once the meta-entity and meta-attribute data has been entered. Meta-relationship data defines important associations between meta-entities and meta-attributes and between meta-relationships and meta-attributes. Meta-entity SYSTEM contains meta-attribute ACCESS_NAME, is an example of a relationship between a meta-entity and a meta-attribute. Further, relationships describing the constraints involving which entity-types can participate in which relationship-types, must also be added to the IRD schema description layer. These constraints were previously defined in Figure 5.6.

E. PHYSICAL IMPLEMENTATION OF IRDS DATA

Before beginning the actual IRDS implementation discussion, a brief description of the relational database model will be presented. This discussion is necessary because the implementation of the NWTDB's IRDS is based on

the relational database model. Similar NWTDB terms concerning the structure of the database are only coincidental and should not be compared with terms relating to the relational database model.

The relational database model is based on the concept that data is stored in two-dimensional tables called relations. Each row in this table represents a record and this record is called a tuple. Each column in this table represents a field and this field is called an attribute. This entire table is roughly equivalent to a file.

Certain restrictions are imposed on these relations. First, attributes are singled valued; neither repeating groups nor arrays are allowed. Second, entries in any column are all of the same kind. For example, one column may contain access names, and another may contain descriptive names. Further, each attribute has a unique name, and attribute positions are insignificant. Finally, no two tuples or rows in a relation may be identical, and the order of tuples is also insignificant [Ref. 14:pp. 132-133].

Since the IRDS is based on the relational database model, the physical implementation of the dictionary system will require the creation of two relations, ENTITY and RELSHIP, and views of these relations. The attribute

structure for each of these relational tables is presented in Figure 5.11 [Ref. 13:p. 53]. Views of these relations were early demonstrated in Figure 5.2 and Figure 5.9.

This relational structure allows for the three logical IRDS layers to be integrated into one physical layer. Data for the IRD, the IRD schema, and the IRD schema description layers are physically located in the ENTITY and RELSHIP tables. An extract of the NWTDB's IRDS relational tables are presented in Figure 5.12a (ENTITY) and Figure 5.12b (RELSHIP).

```
ENTITY (entity_name, entity_type, descriptive_name,
        added_by, date_added, modified_by, date_modified,
        modification_number, duration_value,
        duration_type, comments, description,
        security_classification, language, lines_of_code,
        number_of_tables, number_of_records,
        record_category, data_classification,
        document_category)

RELSHIP (relationship_type, first_entity's_name,
         first_entity's_type, second_entity's_name,
         second_entity's_type, access_method,
         frequency, relative_position)
```

Figure 5.11 IRDS Entity-Relationship Model

	ENTITY NAME	ENTITY TYPE	DESCRIPTIVE NAME	ADDED BY	DATE ADDED
1	DESIG_ACFT	ELEMENT	AIRCRAFT DESIGNATOR	B.A. BROWN	10 OCT 86
2	ELEMENT	ENTITY TYPE	DATA ELEMENT	D.F JONES	
3	CONTAINS	REL_SHIP TYPE	CONTAINS REL_SHIP	B.A BROWN	
4	ANAME	ATTRIBUTE TYPE	ACCESS NAME		
5	ACFT_FILE	FILE			
6	FILE	ENTITY TYPE			
7	TABLE				

Figure 5.12a Extract of ENTITY Table

	RELATIONSHIP TYPE	ENTITY1 NAME	ENTITY1 TYPE	ENTITY2 NAME	ENTITY2 TYPE
1	CONTAINS	ACFT_FILE	FILE	DESIG_ACFT	ELEMENT
2	CONTAINS	FILE	ENTITY TYPE	ELEMENT	
3	PROCESSES	USER	ENTITY TYPE	FILE	
4	RESP_FOR	NIAC	USER		
5	RUNS	USER			
6	CONTAINS	PWR_PLANT			
7	DERIVED FROM				

Figure 5.12b Extract of RELSHIP Table

In order to implement this data into ORACLE, relations and views are created using the SQL CREATE TABLE and CREATE VIEW commands. Figure 5.13 presents the SQL commands needed to create relations and views in the IRDS [Ref. 13:p. 56].

SQL command for the creation of an ENTITY table:

```
CREATE TABLE ENTITY
(ENAME                CHAR(15)   NOT NULL,
 ETYPE                CHAR(10)   NOT NULL,
 DESCRIPTIVE_NAME     CHAR(30),
 ADDED_BY             CHAR(30)   NOT NULL,
 DATE_ADDED           DATE       NOT NULL,
 .                    .
 .                    .
 .                    .
 .                    .
 DATA_CLASS           CHAR(10),
 DOCUMENT_CATEGORY    CHAR(10));
```

SQL command for the creation of an Entity view:

```
CREATE VIEW USER AS
(SELECT ACCESS_NAME, DESCRIPTIVE_NAME, ADDED_BY,
        DATE_ADDED, MODIFIED_BY, DATE_MODIFIED,
        MODIFICATION_NUMBER, COMMENTS,
        DESCRIPTION, SECURITY_CLASSIFICATION
 FROM   ENTITY
 WHERE  ETYPE = 'USER');
```

Figure 5.13 Creating Relations and Views in ORACLE

F. IMPLEMENTATION OF CORE FACILITIES

After the physical implementation of IRDS data, the basic structure of the NWTDB's dictionary system has been completed. ORACLE provides the NWTDB with numerous facilities that are essential for normal operation. This database management system provides the NWTDB with the ability to populate, maintain, or delete any portion of the database, data dictionary, and dictionary schema. ORACLE also provides the NWTDB with the ability to retrieve vital information concerning the relationships of the NWTDB. The NWTDB will also have the ability to retrieve information concerning IRD entities, their associated relationships, and the attributes of these entities and relationships as well as information concerning the IRD schema description. Lastly, ORACLE provides the NWTDB with the ability to import and export data from one IRD to another. Typical SQL queries are presented in Figure 5.14. The standardization of the NWTDB's dictionary enhances the process of transferring data between dictionaries. At the present time, there is no such need for this function. As previously stated, the implementation of the NWTDB's IRDS is almost complete. There are, however, necessary features that need to be added to enhance the capabilities of the NWTDB.

```
SQL query command for IRD schema description information:

          SELECT    E2NAME FROM CONTAINS
          WHERE     E1NAME = 'DOCUMENT' AND
                    E2TYPE = 'ENT_TYPE';

Results:            E2NAME
                    DOCUMENT
                    RECORD
                    ELEMENT


          [ Entity-types contained in DOCUMENT ]


SQL query command for IRD schema information:

          SELECT    ENTITY2_NAME FROM CONTAINS
          WHERE     ENTITY1_NAME = 'ACFT' AND
                    ENTITY1_TYPE = 'FILE' AND
                    ENTITY2_TYPE = "ELEMENTS";

Results:            ENTITY2_NAME
                    NAME_ACFT
                    TYPE_ACFT
                    WT_PAYLOAD_KG
                    .
                    .
                    .
                    SPD_AIRCRAFT_MAX-MACH


            [ Elements contained in ACFT file ]
```

Figure 5.14 Examples of SQL Query Commands

1. Core Security

In order to provide the NWTDB with core security, two new entity-types are added to the IRD schema. The creation of DICTIONARY_USER and VIEW entity-types are necessary for the implementation of this security mechanism [Ref. 3:pp. 83-86]. Once these entity-types exist, their associated entities can be generated.

A DICTIONARY_USER entity is created for each authorized user of the NWTDB's IRDS. This entity establishes specific attributes that uniquely defines the user's level of access. A VIEW entity is created for each different logical partition of the IRD. This entity establishes specific attributes that uniquely define permissions and restrictions to read, add to, modify, and delete the entities that are associated with this view.

Lastly, in order to establish important associations between DICTIONARY_USER and VIEW entities the HAS relationship-type is added to the IRD schema. A specific view will be associated with each individual user of the NWTDB.

2. Versioning Facility

The versioning facility provides a unique version-identifier for each distinct entity in the IRD. The version-identifier consists of a variation name and a revision number. For all instances of IRD entities, a version-identifier will be enclosed in parentheses and

attached to each individual access name and descriptive name [Ref. 3:p. 75]. If, for example, the access name of an entity is NAME_ACFT the version-identifier might be NAME_ACFT (version-1:4). This appendage indicates that the entity is in its first version and that this particular version of that entity is in its fourth revision. As changes are made to the IRD, the versioning facility provides the ability to distinguish between different modifications of the NWTDB's entities.

3. Life-Cycle-Phase and Quality-Indicator Facilities

In order to establish life-cycle-phase and quality-indicator facilities, two new meta-entities are added to the IRD schema [Ref. 3:pp. 76-81]. These meta-entities might be designated as LIFE_ CYCLE_PHASE-TYPE and QUALITY_INDICATOR-TYPE. Once these meta-entities are instituted, associated entity-types will be established.

The life-cycle-phase meta-entity requires three new entity-types: UNCONTROLLED, CONTROLLED, ARCHIVED, and SECURITY which will be integrated into the Core System-Standard Schema structure. The quality-indicator meta-entity does not require a particular entity-type structure. The distinct meta-entity structure of quality-indicators will be determined by the needs and desires of the organization.

The creation of life-cycle-phase and quality-indicator meta-entities enables the NWTDB's IRDS to assign

each individual user a specific view, and each view is associated with a unique life-cycle-phase or quality-indicator.

4. Alternate Name Facility

In the IRDS standard, ALTERNATE NAME was used as an IRD attribute to define one distinct alias for each entity of the NWTDB. This structure limits the ability of the NWTDB to associate a specific access name with multiple aliases. In order to alleviate this situation, a new relationship-type will be added to the Core System-Standard Schema. This relationship-type will be designated as ALIAS. A typical example of the attribute structure of this relationship is presented in Figure 5.15. This new relationship-type allows for each NWTDB entity to be assigned numerous associated aliases.

```
ALIAS (access_name, alternate_name1)

ALIAS ('NAME_ACFT', 'NAME_OF_AIRCRAFT')
ALIAS ('NAME_ACFT', 'ACFT_NAME
ALIAS ('NAME_ACFT', 'AIRCRAFT_NAME'
ALIAS ('NAME_ACFT', 'NAME_OF_ACFT'
ALIAS ('NAME_ACFT', 'AIRPLANE_NAME
ALIAS ('NAME_ACFT', 'NAME_OF_AIRPLANE'
```

Figure 5.15 Entity-Type Structure for ALIAS

G. IRDS MODULES

Once the IRDS foundation is built by implementation of the Core Standard, any additional modules added are considered enhancements and can be implemented incrementally as requirements dictate. The potential benefits of each module must be weighed against the cost in resources it will require to implement and maintain it. The modular nature of this system lends itself nicely to incremental implementation and will reduce the project's development risk. Each phase of the development process can be formally reviewed and adjustments made before proceeding to the next phase. This will reduce the risk of implementing inappropriate or unnecessary modules and preclude the wasting of valuable development resources. It is recommended that in order to obtain a base level of functionality for the passive IRDS the minimum implementation should include the Entity Level Security Module, the Application Program Interface Module, and the Standard Data Model Module.

1. Entity Level Security Module

As an extension to the Core Standard Security Facility, this module provides access control to not just entities of a certain type, but to a particular entity within a given type. The following entity, relationship and attribute types are added to the IRDS structure:

- Entity-types: ACCESS_CONTROLLER, VIEW
- Relationship-type: SECURED_BY
- Attribute-types: READ_LOCK, WRITE_LOCK, READ_KEY, WRITE_KEY

A user secures an entity by establishing a relationship between that entity and an ACCESS_CONTROLLER entity. The READ_LOCK and WRITE_LOCK attributes (associated with the ACCESS_CONTROLLER entity-type) serve to protect that entity from unauthorized access. To gain access to the secured entity, a user's VIEW must have READ_KEY and/or WRITE_KEY attributes (associated with the VIEW entity-type) that match the READ_LOCK and/or WRITE_LOCK attributes of the entities' ACCESS_CONTROLLER [Ref. 3:pp. 105-107]. These attribute-types utilize 10 digit numbers which are assigned and controlled by the IRDS.

2. Application Program Interface Module

This module provides the ability to access the IRDS through external software. The IRDS is basically a subroutine of the application. Communication is established by simply passing IRDS syntactically correct commands and parameters through the application's Command Language Interface call routine. Parameters must exist for receiving output and any error condition returned by the IRDS. Using this module to interface the IRDS with Automated Conversion facilities will alleviate some of the problems of initially populating the IRDS. Programs can be written which will

read and detect metadata from applications, reports, files or any other source of metadata. Many hours of manual effort can be spared if metadata sources are scanned, the data is encoded and then automatically inserted into the IRDS structure by a conversion facility. The ability to analyze metadata can also enhance standardization efforts. These facilities can detect and disclose inconsistencies in the names, formats, and structures used in IRDS applications or even other databases [Ref. 15:pp. 256-258]. The initial conversion of the NWTDB into the IRDS structure should also be automated as much as possible by use of conversion facilities.

3. Standard Data Model Module

An implementation of this module does not change Core Standard functions. A new schema is created in the IRDS which describes an organization's standard database structure. This information can then be utilized for development and maintenance standardization purposes. It can also be utilized to develop what is referred to as an Enterprise Model. The Enterprise Model will describe the overall enterprise in terms of objects (things, activities, events, policies, concepts, etc.) plus the attributes and relationships among these objects. This schema structure will enhance requirements analysis/specification efforts by

describing the environment's operations, and constraints using semantic constructs within the IRDS [Ref. 15:pp. 21-46].

H. ACTIVE IRDS

The IRDS implementation as described so far will only support a passive data dictionary system. Although the administrative capabilities of the traditional DBMS have been greatly enhanced by including the entire information resource environment, a passive IRDS still only documents the environment rather than interacting with it. The active IRDS, on the other hand, will provide the interface necessary for operationally integrating system software components and change the IRDS from a documentation tool to an interactive development and maintenance tool. This will enhance the support of data administration throughout the life cycle of a system. Providing this active link between system components creates an enhanced control and audit mechanism which dramatically increases the utility of the IRDS.

Unlike the passive IRDS which can be implemented without major allocation of additional hardware, software or personnel resources, an implementation of the active IRDS standards will require extensive analysis and programming efforts. Therefore, this system should not be attempted without proper planning, preparation, and allocation of the

necessary resources. The Services Interface standard [Ref. 4] contains all definitions, constants, data types, specifications and service protocols necessary to implement the active IRDS. This will not only simplify the process of programming and coding but acts as a sole source for standardized development of IRDS implementations. An IRDS implementation for the NWTDB paves the way for "activating" NWTDB standards in subsequent programs which will use its metadata.

VI. CONCLUSION

A. OVERVIEW

Tactical system compatibility and interoperability will continue to play a vital role in today's warfighting environment. Tactical information cannot be mishandled, misplaced or misinterpreted. There must be a single, authoritative standard for data elements, databases and communication protocols Navy-wide. Both DoD and DoN are in the process of developing and implementing standardization policies for tactical information systems. Having been selected by the CNO to be the tactical database standard for the Navy, the NWTDB project is a key player in this standardization effort. The NWTDB project must take an innovative and creative leadership role to identify and validate data sources, coordinate and integrate existing tactical information systems, and provide a model for system development emulation. It must also attempt to establish the credibility necessary to be the sole authoritative reference standard for Navy tactical databases.

The Navy is desperately in need of more efficient and effective ways to manage and control its information resources. New data administration tools and techniques are needed to support today's complex information resource environment. Given the critical nature of tactical

information and the problems associated with standardization, data administration systems such as the IRDS have become a necessity for the Navy. Capabilities for centrally managing all aspects of an information system can be satisfied through the IRDS concept. The NWTDB must be upgraded to provide IRDS and, therefore, expanded data administration capabilities for tactical data. The potential benefits from implementing the IRDS should outweigh the relatively modest investment a system such as this will require.

B. PASSIVE IRDS

Implementation of the IRDS expands the management scope of control to include the entire information resource environment. The system is changed from a collection of segregated and independent components, to a single cohesive system. The information system can now be managed centrally, creating a globally administered and standardized resource environment. This will have far reaching implications for developing and enforcing standards, developing requirements, assessing the impact of changes, and controlling the integrity and security of the information system. Since the IRDS will have the ability to maintain information on the entire information system, many development and maintenance tools will no longer need to maintain their own proprietary data. This will eliminate

the need to maintain redundant data, fragmented information and disparate descriptions of information resources.

The IRDS will not just be a repository for data elements. It will also contain a description of the NWTDB and all the systems and applications interacting with it. The IRDS schema will be an invaluable tool utilized for not only data administration, but for many different support functions, such as planning, analysis, design, construction and maintenance of information resources. The IRDS will also be a vehicle for the standardization and integration of heterogeneous databases within the Navy. Given that the IRDS concept is accepted and successfully implemented for the NWTDB, the IRDS standard will then become the Navy standard format for tactical database systems. The integration and interoperability of systems conforming to the same IRDS standards can then be easily accomplished.

A database system such as the NWTDB will be progressively more difficult to maintain over its life cycle. As the system becomes larger and more complicated, maintenance costs will steadily grow, patches will be less and less effective and structural changes will become essentially impossible to implement. The documentation capabilities of the IRDS will greatly enhance and facilitate maintenance operations. The IRDS schema will provide a unique source for anticipating the side effects caused by structural changes. The effective utilization of these

maintenance enhancements will keep the system responsive to its changing environment.

C. ACTIVE IRDS

As mentioned already, the IRDS documents data utilized by external applications, programs or other systems operating in the same domain as the IRDS. The active IRDS takes this one step further by allowing those external resources to interactively access and maintain the IRDS. This will eliminate the need to manually update the IRDS every time a change is made to a data element or structure within the IRDS. It will insure that data definitions being used by any system resource will be the most current and correct data available. This interactive system will allow end users to query and update the IRDS from within their resident programs. IRDS functions and facilities will be readily available to system analysts, system designers, programmers, data administrators, database managers or any other user needing access to the IRDS.

> The true value of an IRDS exhibits itself when the IRDS can be included in an information processing loop as an active control and audit mechanism: for example, at the front end of a DBMS, data design,, or requirements analysis software system. Establishing active links between the IRDS and operational information systems dramatically increases the utility of an IRDS. [Ref. 13:p. 59]

A globally controlled support system is the key to integration of analysis facilities with software development, programming with application environments and

finally, management with all components. The last few years has seen tremendous improvements in the area of automated software development. Computer Aided Software Engineering (CASE) tools provide not only facilities for the analysts and systems developers (dataflow diagrams, entity-relationship diagrams, logical data models, state-transition diagrams, transformation graphs and decision matrices), but also features valuable to project management (consistency, completeness, conformance, reliability and documentation functions). The integration of CASE tools with the IRDS will provide an on-line reference library to support development and maintenance throughout the system's life cycle. Up-to-date data stored in the IRDS can easily be evaluated against the systems requirements, design, implementation and maintenance records.

A database "workbench" environment is seen as the ultimate system of the future. Supported by the IRDS, the workbench will be a collection of independent tools (e.g. CASE tools, application generators, input and edit utilities, compilers, report writers, etc.) all being able to communicate with the IRDS and one another. Each tool will support specific phases of the system life cycle, relying on the IRDS for inter-tool communication and for storage of information common to all tools. A way to store and manage all the components of application development in one dictionary system is the cornerstone for automation,

reusability and productivity in the future [Ref. 16:p. 42]. The active IRDS is the logical foundation upon which a system such as this can be built.

D. RECOMMENDATIONS

This thesis has presented an overview of the IRDS and active IRDS models and related these concepts to the NWTDB system. The importance and applicability of the IRDS to the Navy's control and management of information resources is clear. Providing the tactical environment with compatible and interoperable systems requires the use of sophisticated DA tools such as the IRDS. We believe that a phased, incremental IRDS implementation, carefully evaluated at each stage, is a critical step in the development of an integrated Navy tactical database environment.

LIST OF REFERENCES

1. OPNAVINST 9410.XX Encl (1) (Draft Proposed), NWTDB Concept of Operations.

2. Naval Intelligence Automation Center, Department of Defense, Naval Warfare Tactical Data Base: User's Guide, Planning Research Co., McLean, VA, January 1988.

3. U.S. Department of Commerce National Bureau of Standards, Report NBSIR 88-3700, A Technical Overview of the Information Resource Dictionary System, by A. Goldfine and P. Konig, January 1988.

4. American National Standard for Information Systems (Draft Proposed), Report X3.nnn-1988, Information Resource Dictionary System-Services Interface, by American National Standards Institute, Inc., 1988.

5. Chief of Naval Operations letter 7110: Ser 942/8U539107 to the Department of the Navy, Subject: Naval Warfare Tactical Data Base (NWTDB) Policy and Implementation Guidance, 6 April 1988.

6. OPNAVINST 9410.XX (Draft Proposed), Naval Warfare Tactical Data Base (NWTDB).

7. Naval Intelligence Automation Center, "Data Dictionary Information," Unofficial documented requested from NAVDAC (code 10).

8. Martin, J. and McClure, C., "The Maintenance Mess," Software World, v. 14, August 1983.

9. Allen, F., Loomis, M. and Mannino, M., "The Integrated Data Dictionary/Directory System," Computing Surveys, v. 14, June 1982.

10. DOD Directive 5000.11, Data Elements and Data Codes Standardization Program, October 1988.

11. Chief of Naval Operations: Ser 09/8U501009 to the Department of the Navy, Subject: Tactical Naval Warfare Systems Data Base and Communications Format Standardization Policy, 5 July 1988.

12. Pressman, R. S., Software Engineering: A Practitioner's Approach, 2nd ed., McGraw-Hill Book Co., 1987.

13. Dolk, D. R. and Kirsch, R. A. II, "A Relational Information Resource Dictionary System," Computing Practices, v. 30, January 1987.

14. Kroenke, D. M. and Dolan, K. A., Database Processing, 3rd ed., Science Research Associates, Inc., 1988.

15. Navathe, S. B. and Kerschberg, L., "Role of Data Dictionaries in Information Resource Management," Information and Management, v. 10, January 1986.

16. Acly, E., "Looking Beyond CASE," IEEE Software, March 1988.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Naval Intelligence Automation Center Mr. Al Watson, NIAC-821 4600 Silver Hill Road Washington, DC 20389-5000	1
4.	Department of the Navy Office of the Chief of Naval Operations OPNAV-945, LCDR Black Washington, DC 20350-2000	1
5.	Naval Data Automation Command Code 42, LT Lubinsky Washington Navy Yard Washington, DC 20374-1662	1
6.	Professor Dan Dolk, Code-54DK Department of Administrative Sciences Naval Postgraduate School Monterey, CA 93940-5000	2
7.	LT Bruce A. Brown 2043 Alcazar Way Stockton, CA 95207	2
8.	LT Todd VanGunten 601 East Circle Drive Findlay, OH 45840	2

Thesis
B80975 Brown
c.1 Naval Warfare Tactical
Data Base.